CA Gen

User Exit Reference Guide

Release 8.5

Third Edition

CA Technologies Product References

This document references the following CA Technologies products:

- CA Gen

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- ABRT_xcall_ws_url_exit —CALL EXTERNAL Web Service URL Exit (see page 817)—Added new user exit.
- ABRT_xcall_ws_gentype_truncate_exit —CALL EXTERNAL Data Truncation (see page 744)—Added new user exit.
- ServerError Method—Detects an error during the processing of a synchronous client to server flow (see page 669)-Added a new parameter description.
- WebServiceMethodCallExit (see page 829)-Added new user exit.

Contents

Chapter 1: Introduction 19

Overview 19
User Exits 19
Visual Studio Support 19
64-bit Windows Support 20
User Exits Collections 20
User Exit General Information 21

Chapter 2: Windows C User Exits 25

Windows Blockmode User Exits 26
- DBCOMMIT—Database Commit Exit 27
- DBCONNCT—Database Connection User Exit 29
- DBDISCNT—Database Disconnect Exit 32
- TIRDLCT—Dialect Exit 33
- TIRDRTL—Default Retry Limit Exits 35
- TIRHELP—Help Interface Exit 37
- TIRMTQB—Message Table Exit 40
- TIRSECR—Security Check Interface Exit 43
- TIRSYSID—System ID Exit 46
- TIRTERMA—User Termination Exit 47
- TIRTIAR—Database Error Message Exit 49
- TIRUPDB—MBCS Uppercase Translation Exit 50
- TIRUPPR—Uppercase Translation Exit 52
- TIRURTL—Ultimate Retry Limit Exit 54
- TIRUSRID—User ID Exit 55
- TIRYYX—Date Exit 57

Windows GUI Client User Exits 58
- C4COMMIT—Database Commit Exit (Windows) 60
- C4CONNECT—Database Connection Exit (Windows) 61
- C4DISCONNECT—Database Disconnect Exit (Windows) 63
- C4ERRMSG—Database Message Exit (Windows) 65
- C4ROLLBACK—Database Rollback Exit (Windows) 66
- C4SQLLEN—Database SQLCA Length Exit (Windows) 68
- WRASYNCSRVERROR—Asynchronous Flow Server Failure Exit (Windows) 69
- WRDEFAULTYEAR—Century Default Exit (Windows) 73
- WRDRTL—Default Retry Limit Exit (Windows) 74

WRGLB—Globalization Exit (Windows) 76
WRSECDECRYPT—Client Decryption Exit (Windows) 78
WRSECENCRYPT—Client Side Encryption Exit (Windows) 80
WRSECTOKEN—Client Security Token Exit (Windows) 83
WRSRVRERROR—Server Flow Error Exit (Windows) 87
WRSTRNCM—String Comparison Exit (Windows) 90
WRSYSID—System ID Exit (Windows) 91
WRTERMID—Terminal ID Exit (Windows) 93
WRUPPR—Uppercase Translation Exit (Windows) 94
WRURTL—Ultimate Retry Limit Exit (Windows) 96
WRUSRID—User ID Exit (Windows) 97
Windows Client Middleware User Exits 99
Client Manager - Windows User Exits 99
Communications Bridge - Windows User Exits 123
Common System Utilities - Windows User Exits 142
TCP/IP - Windows User Exits 144
WebSphere MQ Client Transport - Windows User Exits 150
ECI - Windows User Exits 163
Tuxedo 167
Web Services - Windows User Exits 175
Windows Servers User Exits 179
Windows Server Middleware User Exits 199
Windows C Proxy User Exits 205

Chapter 3: UNIX and Linux User Exits 215

UNIX and Linux Blockmode User Exits 216
DBCOMMIT—Database Commit Exit 217
DBCONNCT—Database Connection User Exit 218
DBDISCNT—Database Disconnect Exit 220
TIRDLCT—Dialect Exit 221
TIRDRTL—Default Retry Limit Exits 222
TIRHELP—Help Interface Exit 224
TIRMTQB—Message Table Exit 227
TIRSECR—Security Check Interface Exit 230
TIRSYSID—System ID Exit 232
TIRTERMA—User Termination Exit 234
TIRTIAR—Database Error Message Exit 235
TIRUPDB—MBCS Uppercase Translation Exit 237
TIRUPPR—Uppercase Translation Exit 239
TIRURTL—Ultimate Retry Limit Exit 240
TIRUSRID—User ID Exit 241

TIRYYX—Date Exit 243
UNIX/Linux Client Middleware User Exits 244
Common System Utilities - UNIX and Linux User Exits 244
TCP/IP - Windows User Exits 246
WebSphere MQ Client Transport - Windows User Exits 252
Tuxedo 264
UNIX and Linux Server User Exits 272
SRVRERROR—Server to Server Error Exit 273
TIRDCRYP—Server Decryption Exit 276
TIRELOG—Server Error Logging and Error Token Creation Exit 278
TIRNCRYP—Server Encryption Exit 280
TIRSECV—Security Validation Exit 282
TIRXINFO—Locale Information Exit 285
TIRXLAT—National Language Translation Exit 287
UNIX and Linux Asynchronous Daemon User Exits 290
AEFSECEX—Asynchronous Daemon Security Exit 290
UNIX/Linux Server Middleware User Exits 292
WebSphere MQ Server Transport - UNIX and Linux User Exits 292
Tuxedo - UNIX and Linux User Exits 298
CI_S_POST_END—Tuxedo After Transaction Termination Exit 299
CI_S_POST_SVRDONE—Tuxedo After Server Shutdown Exit 300
CI_S_POST_SVRINIT—Tuxedo After Server Initialization Exit 301
CI_S_POST_BEGIN—Tuxedo After Begin Transaction Exit 303
CI_S_PRE_END—Tuxedo Prior to Transaction Termination Exit 304
CI_S_USER_DATA_IN—Tuxedo Inbound Flow Data Access Exit 306
CI_WS_DPC_URL_Exit — Web Services DPC URL User Exit 307
CI_S_USER_DATA_OUT—Tuxedo Outbound Flow Data Access Exit 309
Web Services - UNIX and Linux User Exits 310
CI_WS_DPC_URL_Exit—Web Services DPC User Exit 311
UNIX and Linux C Proxy User Exits 313
WRSECTOKEN—Client Security Token Exit 313
WRSECENCRYPT—Client Side Encryption Exit 317
WRSECDECRYPT—Client Decryption Exit 320

Chapter 4: z/OS User Exits 323

Changes to User Exits 323
z/OS Blockmode User Exits—CICS 325
TIRCDPTX—Dynamic Plan TSQ Processing Exit 325
TIRCRTRX—Default Retry Limit Exit 327
TIRTIARX—DB2 Message Exit 329
TIRCURTX—Ultimate Retry Limit Exit 331

TIRSYSIX—System ID Exit 332
TIRUSRIX—User ID Exit 334
TIRSECRX—Security Interface Exit 336
TIRQCNTX—TSQ Profile Manager Exit 339
TIRDATX—Date and Time Services Exit 341
TIRDEVC—Device Characteristics Exit 347
TIRDLCTX—User Dialect Exit 351
TIRUPPRX—Uppercase Translation Exit 352
TIRYYX—Two-Digit Year Input Edit Exit 354
TIRTERMA—Termination Exit 356
TIRHELPX—Help Interface Exit 364
TIRIEX—Enhanced Map Input Edit Exit 367
TIRIEXS – Standard Map Input Edit Exit 368
z/OS Blockmode User Exits—IMS 375
TIRTIARX—DB2 Message Exit 375
TIRIRTRX—Default Retry Limit Exit 377
TIRIURTX—Ultimate Retry Limit Exit 379
TIRSYSIX—System ID Exit 381
TIRUSRIX—User ID Exit 382
TIRSECRX—Security Interface Exit 384
TIRDATX—Date and Time Services Exit 387
TIRDEVI—Device Characteristics Exit 394
TIRDLCTX—User Dialect Exit 398
TIRUPPRX—Uppercase Translation Exit 400
TIRYYX—Two-Digit Year Input Edit Exit 402
TIRTERMA—Termination Exit 403
TIRMTQB—Runtime Message Table Exit 411
TIRIDTRX—IMS Server Debug LTERM 413
TIRIEX—Enhanced Map Input Edit Exit 414
TIRIEXS – Standard Map Input Edit Exit 415
z/OS Blockmode User Exits—TSO 422
TIRTIARX—DB2 Message Exit 422
TIRIRTRX—Default Retry Limit Exit 424
TIRIURTX—Ultimate Retry Limit Exit 426
TIRSYSIX—System ID Exit 428
TIRUSRIX—User ID Exit 429
TIRSECRX—Security Interface Exit 431
TIRDATX—Date and Time Services Exit 434
TIRDEVT—Device Characteristics Exit 440
TIRDLCTX—User Dialect Exit 444
TIRUPPRX—Uppercase Translation Exit 446
TIRYYX—Two-Digit Year Input Edit Exit 447

TIRMTQB—Runtime Message Table Exit 449
TIRTERMA—Termination Exit 451
TIRIEX—Enhanced Map Input Edit Exit 458
TIRIEXS – Standard Map Input Edit Exit 459
TIRIURTX—Ultimate Retry Limit Exit 466
z/OS Middleware User Exits—CICS TCP/IP Direct Connect Exits 468
TIRSLEXT—CICS Sockets Server Listener Exit 469
TIRSLTMX—CICS Sockets Server Listener TIMEOUT Exit 473
z/OS Middleware User Exits—IMS TCP/IP Direct Connect Exits 477
TIRxxTD—TCP/IP Destination ID Exit 477
TIRxxTDC—TCP/IP Decryption Exit 480
TIRxxTSC—TCP/IP Security Exit 482
z/OS Middleware User Exits – WebSphere MQ CICS 486
WebSphere MQ Transaction Dispatcher for CICS (TDC) Exit 486
z/OS Server User Exits—CICS 490
TIRTIARX—DB2 Message Exit 490
TIRCDPTX—Dynamic Plan TSQ Processing Exit 493
TIRSRTRX—Default Retry Limit Exit Processing 495
TIRSURTX—Server Ultimate Retry Limit Exit 495
TIRSYSIX—System ID Exit 495
TIRUSRIX—User ID Exit 497
TIRSECRX—Security Interface Exit 498
TIRQCNTX—TSQ Profile Manager Exit 501
TIRUPPRX—Uppercase Translation Exit 503
TIRXINFO—National Language Information Exit 505
TIRSECVX—Server Client Security Validation Exit 506
TIRDCRYX—Server Decryption Exit 509
TIRNCRYX—Server Encryption Exit 513
TIRELOGX—Server Error Logging and Error Token Creation Exit 516
TIRALLOX—Server-to-Server Allocate Conversation Exit 518
TIRPTOKX—Server-to-Server Security Token CA Generation Exit 521
TIRCSGNX—Server TCP/IP Signon Exit 523
TIRPROUX—Server-to-Server Routing Exit 525
TIRSIPEX—CICS Sockets Server Exit 527
TIRMQPX—MQ SERIES Put Function Exit 530
z/OS Server User Exits—IMS 532
TIRTIARX—DB2 Message Exit 532
TIRALLOX—Server-to-Server Allocate Conversation Exit 534
TIRPROUX—Server-to-Server Routing Exit 536
z/OS Batch User Exits 539
TIRTIARX—DB2 Message Exit 539
TIRBRTRX—Default Retry Limit Exit 541

TIRBURTX—Ultimate Retry Limit Exit....544
TIRRETCX—Batch Return Code Override Exit....545
TIRTERBX—Batch Termination Exit....546
Customizing and Installing z/OS User Exits....549
MKUEXITS—Make COBOL Runtimes (User Exits DLLs)....550
MKCRUN—Make C Runtimes - TIRCRUNC (CICS) and TIRCRUNI (IMS)....551
MKUECTCP—Make CICS TCP/IP Exits (TIRSLEXT and TIRSLTMX)....553
MKUEITCP—Make IMS TCP/IP Exits (TIRxxTD, TIRxxTDC, and TIRxxTSC)....554

Chapter 5: NonStop User Exits 557

NonStop Blockmode User Exits....557
TIRDLCT—Dialect Exit....558
TIRDRTL—Default Retry Limit Exits....560
TIRHELP—Help Interface Exit....562
TIRMTQB—Message Table Exit....565
TIRSECR—Security Check Interface Exit....569
TIRSYSID—System ID Exit....572
TIRTERMA—User Termination Exit....574
TIRUPDB—MBCS Uppercase Translation Exit....576
TIRUPPR—Uppercase Translation Exit....578
TIRURTL—Ultimate Retry Limit Exit....579
TIRUSRID—User ID Exit....581
TIRYYX—Date Exit....583
NonStop Server User Exits....585
TIRDCRYP—Server Decryption Exit....586
TIRELOG—Server Error Logging and Error Token Creation Exit....589
TIRNCRYP—Server Encryption Exit....592
TIRSECV—Security Validation Exit....595
TIRXINFO—Locale Information Exit....598
TIRXLAT—National Language Translation Exit....600
USEREXIT—NonStop RSC/MP Distributed Processing Flow Data Access Exit....603

Chapter 6: Web Generation User Exits 611

CompareExit—Web Generation Compare Exit....611
Source Code....611
Purpose....611
CompareTo Method—Compares Two Decimals....611
CompareTo Method—Compares Two Characters....612
CompareTo Method—Compares Two Doubles....612
CompareTo Method—Compares Two Floats....613
CompareTo Method—Compares Two Integers....614

CompareTo Method—Compares Two Longs 614
CompareTo Method—Compares Two Objects 615
CompareTo Method—Compares Two Shorts 616
CompareTo Method—Compares Two Strings 617
CompareTo Method—Compares two strings(upto the indicated length) 617
CompareTo Method—Compares Two DateTime instances 618
Rebuilding the Exit 619
DataConversionExit–Web Generation Data Conversion Exit 620
Source Code 620
Purpose 620
modifyInputString Method—Modifies Input String 620
modifyOutputString Method—Modifies Output String 620
Rebuilding the Exit 621
LowerCaseExit– Web Generation Lower Case Exit 622
Source Code 622
Purpose 622
LowerCase Method—Converts String to Lower Case 622
Rebuilding the Exit 623
UpperCaseExit–Web Generation Upper Case Exit 623
Source Code 623
Purpose 623
UpperCase Method—Converts string to Upper Case 623
Rebuilding the Exit 624
EJBRMIContextExit–Web Generation EJB RMI Context Exit 624
Source Code 624
Purpose 625
getInstance Method—Retrieves an instance of the exit class 625
Rebuilding the Exit 625
EJBRMIDynamicCoopFlowExit–Web Generation EJB RMI Dynamic Coop Flow Exit 626
Source Code 626
Purpose 626
getInstance Method—Retrieves an instance of EJBRMIDynamicCoopFlowExit class 626
FreeInstance Method—De-allocates the object obtained with GetInstance() 627
ProcessException Method—Indicates whether to retry the operation or to throw an exception 628
init Method—Initializes the current instance internally from the GetInstance () 629
getInitialFactory Method—Retrieve the initial factory classname 630
getProviderURL Method—Retrieves the providerURL 630
getUserObject Method—Retrieves a User Object 630
Rebuilding the Exit 631
EJBRMISecurityExit–Web Generation EJB RMI Security Exit 631
Source Code 631
Purpose 631

getInstance Method—Allocates a security object that contains all of the security information 632
FreeInstance Method—De-allocates the object obtained with GetInstance 632
validate Method—Verifies the security object is correct 633
getObject Method—Passess the original security object to a Server 633
SecurityType Property—Specifies the type of security 634
Rebuilding the Exit 634
TCPIPDynamicCoopFlowExit–Web Generation TCPIP Dynamic CoopFlow Exit 635
Source Code 635
Purpose 635
getInstance Method—Obtains an instance of TCPIPDynamicCoopFlowExit class 635
FreeInstance Method—De-allocates the object obtained with GetInstance 636
ProcessException Method—Indicates whether to retry the operation or to throw an exception 637
init Method—Initializes the current instance internally from the GetInstance 638
getHostName Method—Retrieves the hostname 639
getPort Method—Retrieves the port 639
geClientPersistence Method—Retrieves the client socket connection persistence state 640
Rebuilding the Exit 640
WindowManagerCfgExit–Web Generation Window Manager Configuration 641
Source Code 641
Purpose 641
URL mapURL Method—Maps the passed URL, load module name and procedure step name 641
Rebuilding the Exit 642
ContextLookupExit–Web Generation Context Look Up 642
Source Code 642
Purpose 642
lookup Method—Retrieves an instance of the named context object 642
Rebuilding the Exit 643
CFBDynamicMessageSecurityExit–Web Generation CFB Dynamic Message Security Exit 643
Source Code 643
Purpose 644
CFBDynamicMessageSecurityExit Constructor—Provides the default caching mechanism 644
GetInstance Method—Obtains an instance of CFBDynamicMessageSecurityExit class 644
FreeInstance Method—De-allocates the object obtained with GetInstance 645
getSecurityToken Method—Allows the user to pass back a security token 646
Init Method—Initializes the current instance internally from the GetInstance 646
getSecurityType Method—Specifies the type of security 647
useCMSecurity Method—Specifies whether the Client Manager/Comm Bridge to use the userID and password 648
Rebuilding the Exit 648
CFBDynamicMessageEncodingyExit–Web Generation CFB Dynamic Message Encoding Exit 649
Source Code 649
Purpose 649

serverEncoding Method—Retrieves the message text encoding for the named host and transaction 649
Rebuilding the Exit 650
CFBDynamicMessageEncryptionExit–Web Generation CFB Dynamic Message Encryption Exit 650
Source Code 650
Purpose 650
CFBDynamicMessageEncryptionExit Constructor—Provides the default caching mechanism 650
GetInstance Method—Obtains an instance of CFBDynamicMessageEncryptionExit class and initializes it 651
FreeInstance Method—De-allocates the object obtained with GetInstance 652
encryptData Method—Allows the user to encrypt the data portion of the message 652
Init Method—Initializes the current instance internally from the GetInstance.....653
Rebuilding the Exit 654
CFBDynamicMessageDecryptionExit–Web Generation CFB Dynamic Message Decryption 654
Source Code 654
Purpose 654
CFBDynamicMessageDecryptionExit Constructor—Provides the default caching mechanism.....654
Method 655
FreeInstance Method—De-allocates the object obtained with GetInstance 655
decryptData Method—Decrypts the data portion of the message 656
Init Method—Initializes the current instance internally from the GetInstance.....657
doDecryption Method—Specifies whether decryption should be done 657
Rebuilding the Exit 658
DefaultYearExit–Web Generation Default Year Exit 658
Source Code 658
Purpose 658
GetDefaultYear Method—Implements a customer-specified algorithm addressing Year-2000 concerns.....659
Rebuilding the Exit 659
LocaleExit–Java Locale Exit.....660
Source Code 660
Purpose 660
getLocalCurrencySymbol Method—Supplies the currency symbol to the generated JAVA application 660
getLocalThousandsSep Method—Supplies the thousand separator to the generated JAVA application 661
getLocalDecimalSeparator Method—Supplies the decimal point to the generated JAVA application 661
getLocalDateSeparator Method—Supplies the date separator character to the generated JAVA application.....662
getLocalTimeSep Method—Supplies the time separator character to the generated JAVA application 663
getLocalDateOrder Method—Supplies the date order definition to the generated JAVA application 663
Rebuilding the Exit 664
RetryLimitExit–Web Generation Retry Limit Exit 665
Source Code 665
Purpose 665

getUltimateRetryLimit Method—Retrieves the Integer containing absolute upper limit to the number of times a procedure step can be retried....665
getDefaultRetryLimit Method—Retrieves the Integer containing default retry limit number of times a procedure step can be retried....666
Rebuilding the Exit....666
SessionIDExit–Web Generation Session ID Exit....666
Source Code....666
Purpose....667
getSystemId Method—Retrieves the String containing the value for the LOCAL_SYSTEM_ID attributes....667
getUserId Method—Retrieves the String containing the value for the USER_ID attributes....667
getTerminalId Method—Retrieves the String containing the value for the TERMINAL_ID attributes....668
Rebuilding the Exit....668
SrvrErrorExit–Web Generation Server Error Exit....669
Source Code....669
Purpose....669
ServerError Method—Detects an error during the processing of a synchronous client to server flow....669
append Method—Formats errors with messages unique to your application....671
Method....672
Rebuilding the Exit....672
UserExit–Web Generation User Exit....673
Source Code....673
Purpose....673
Default Behavior....673
startUp Method—Instantiates the UserExit class with its properties initialized....673
getCurrencySign Method—Retrieves the currency sign value for the current UserExit object....675
getThousandsSeparator Method—Retrieves the Thousand Separator value for the current UserExit object....675
getDecimalSeparator Method—Retrieves the Decimal Separator value for the current UserExit object....676
getDateSeparator Method—Retrieves the Date Separator value for the current UserExit object....676
getTimeSeparator Method—Retrieves the Time Separator value for the current UserExit object....676
getDateOrder Method—Retrieves the Date Order value for the current UserExit object....677
getMessageFile Method—Retrieves the two letter key to select the message file....677
getSystemId Method—Retrieves the system ID string attribute....678
getUserId Method—Retrieves the userID string attribute....678
getTerminalId Method—Retrieves the terminal ID string attribute....679
getDialectName Method—Retrieves the current dialect name for the load module....679
GetDefaultYear Method—Implements a customer-specified algorithm addressing Year-2000 concerns....679
padAndTrim Method—Trims and pads the given string with the specified arguments....680
Rebuilding the Exit....681
WSDynamicCoopFlowExit–Web Service Dynamic Coop Flow Exit....681
Source Code....681
Purpose....681

getInstance Method—Retrieves an Instance of WSDynamicCoopFlowExit Class 682
freeInstance Method—De-allocates the Object Obtained with GetInstance() 683
processException Method—Indicates Whether to Retry the Operation or to Throw an Exception 684
init Method—Initializes the Current Instance Internally from the getInstance () 685
getBaseURL Method—Retrieves the baseURL 686
getContextType Method—Retrieves the contextType 686

Chapter 7: Web View User Exits 689

WVDefaultYearExit–WebView Default Year Exit 689
Source Code 689
Purpose 689
Rebuilding the Exit 689
WVLocaleExit–WebView Locale Exit 689
Source Code 689
Purpose 690
getLocalCurrencySymbol Method—Supplies the currency symbol to the generated JAVA application 690
getLocalThousandsSep Method—Supplies the thousand separator to the generated JAVA application 690
getLocalDecimalSeparator Method—Supplies the decimal point to the generated JAVA application 691
getLocalDateSeparator Method—Supplies the date separator character to the generated JAVA application 692
getLocalTimeSep Method—Supplies the time separator character to the generated JAVA application 692
getLocalDateOrder Method—Supplies the date order definition to the generated JAVA application 693
Rebuilding the Exit 694
WVRetryLimitExit–WebView Retry Limit Exit 695
Source Code 695
Purpose 695
getUltimateRetryLimit Method—Retrieves the Integer containing absolute upper limit 695
getDefaultRetryLimit Method—Retrieves the Integer containing default retry limit 695
Rebuilding the Exit 696
WVSessionIDExit–WebView Session ID Exit 696
Source Code 696
Purpose 696
getSystemId Method—Retrieves the String containing the value for the LOCAL_SYSTEM_ID attributes 697
getUserId Method—Retrieves the String containing the value for the USER_ID attributes 697
getTerminalId Method—Retrieves the String containing the value for the TERMINAL_ID attributes 697
Rebuilding the Exit 698
WVSrvrErrorExit–WebView Server Error Exit 698
Source Code 698
Purpose 699
ServerError Method—Detects an error during the processing of a synchronous client to server flow 699
Method 700

Method 701
Rebuilding the Exit 702
WVUserExit–WebView User Exit 702
Source Code 702
Purpose 702
Default Behavior 702
startUp Method—Instantiates the UserExit class with its properties initialized 703
getCurrencySign Method—Retrieves the currency sign value for the current UserExit object 704
getThousandsSeparator Method—Retrieves the Thousand Separator value for the current UserExit object 705
getDecimalSeparator Method—Retrieves the Decimal Separator value for the current UserExit object 705
getDateSeparator Method—Retrieves the Date Separator value for the current UserExit object 705
getTimeSeparator Method—Retrieves the Time Separator value for the current UserExit object 706
getDateOrder Method—Retrieves the Date Order value for the current UserExit object 706
getMessageFile Method—Retrieves the two letter key to select the message file 706
getSystemId Method—Retrieves the system ID string attribute 707
getUserId Method—Retrieves the user ID string attribute 708
getTerminalId Method—Retrieves the terminal ID string attribute 708
getDialectName Method—Retrieves the current dialect name for the load module 708
GetDefaultYear Method—Implements of a customer-specified algorithm addressing Year-2000 concerns 709
padAndTrim Method—Trims and pads the given string with the specified arguments 709
Rebuilding the Exit 710

Chapter 8: .NET User Exits 711

ASP.NET Web Client User Exits 711
com.ca.gen.exits.amrt.DefaultYearExit–C# Default Year Exit 711
com.ca.gen.exits.amrt.LocaleExit–C# Locale Exit 712
com.ca.gen.exits.amrt.RetryLimitExit–C# Retry Limit Exit 717
com.ca.gen.exits.amrt.SessionIdExit–C# Session ID Exit 718
com.ca.gen.exits.amrt.SrvrErrorExit–C# Server Error Exit 720
com.ca.gen.exits.amrt.UserExit–C# User Exit 724
Exits in <CAGen-root>\.net\exits\src\Common.cs file Used by CA Gen ASP.NET Web Clients and CA Gen .NET Servers 732
com.ca.gen.exits.common.CompareExit – C# Compare Exit 732
com.ca.gen.exits.common.LowerCaseExit – C# Lower Case Exit 741
com.ca.gen.exits.common.UpperCaseExit – C# Upper Case Exit 742
com.ca.gen.exits.common.WebServiceMethodCallExit- C # CALL EXTERNAL User Exit 743
Exits in <CAGen-root>\.net\exits\src\msgobj Directory Used by CA Gen ASP.NET Web Clients and CA Gen .NET Servers 747
com.ca.gen.exits.msgobj.cfb.CFBDynamicMessageDecryptionExit – C# CFB Dynamic Message Decryption 748

com.ca.gen.exits.msgobj.cfb.CFBDynamicMessageEncodingyExit – C# CFB Dynamic Message Encoding Exit....751
com.ca.gen.exits.msgobj.cfb.CFBDynamicMessageEncryptionExit – C# CFB Dynamic Message Encryption Exit....753
com.ca.gen.exits.msgobj.cfb.CFBDynamicMessageSecurityExit – C# CFB Dynamic Message Security Exit....757
CA Gen .NET Servers....762
com.ca.gen.exits.scrt.AuthorizationExit – C# Server Authorization Exit....762
com.ca.gen.exits.scrt.SecurityValidationExit – C# Server Security Validation Exit....764
com.ca.gen.exits.scrt.LocaleExit – C# Server Locale Exit....767
com.ca.gen.exits.scrt.RetryLimitExit – C# Server Retry Limit Exit....771
com.ca.gen.exits.scrt.SrvrErrorExit – C# Server Error Exit....772
com.ca.gen.exits.scrt.UserExit – C# Server User Exit....774
C# Server Middleware User Exits....777
com.ca.gen.exits.coopflow.net.NETDynamicCoopFlowExit–C# NET Dynamic CoopFlow Exit....777
Exits in <CAGen-root>\.net\exits\src\coopflow directory Used by CA Gen ASP.NET Web Clients and CA Gen .NET Servers....783
com.ca.gen.exits.coopflow.complus.COMPLUSDynamicCoopFlowExit–C# COM PLUS Dynamic Coop Flow Exit....783
com.ca.gen.exits.coopflow.complus.COMPLUSDynamicCoopFlowSecurityExit–C# COMPLUS Dynamic Coop Flow Security Exit....788
com.ca.gen.exits.coopflow.mqs.MQSDynamicCoopFlowExit–C# MQSeries Dynamic Coop Flow Exit....791
com.ca.gen.exits.coopflow.net.NETDynamicCoopFlowSecurityExit–C# NET Dynamic Coop Flow Security Exit....799
com.ca.gen.exits.coopflow.tcpip.TCPIPDynamicCoopFlowExit–C# TCPIP Dynamic Coop Flow Exit....802
com.ca.gen.exits.coopflow.ws.WSDynamicCoopFlowExit – C# WS Dynamic Coop Flow Exit....808

Chapter 9: Browser User Exits 815

Customize userOnLoad in ASP.NET Mode....815
Customize userOnLoad in HTML Mode for Web View....816

Chapter 10: Action Block Runtime User Exit 817

Windows Action Block Runtime User Exits....817
ABRT_xcall_ws_url_exit —CALL EXTERNAL Web Service URL Exit....817
ABRT_xcall_ws_gentype_truncate_exit —CALL EXTERNAL Data Truncation....819
UNIX and Linux Action Block Runtime User Exits....823
ABRT_xcall_ws_url_exit —CALL EXTERNAL Web Service URL Exit....823
ABRT_xcall_ws_gentype_truncate_exit —CALL EXTERNAL Data Truncation....825
Java Action Block Runtime User Exits....829
WebServiceMethodCallExit....829

Chapter 1: Introduction

Overview

All the user exits in CA Gen are grouped in this section.

This section is intended for programmers and application developers working on building applications. It is assumed that these users have knowledge about the set of user exits that are invoked from the execution environments in which their applications' components are deployed.

User Exits

CA Gen generated applications invoke a number of CA Gen supplied routines at execution time to perform various functions. Most of these routines are provided as DLLs or shared libraries and cannot be modified by the user. However, some routines have been implemented as user exits.

A user exit is modifiable source code that you can customize to fit your specific needs. The user exits can be as simple or as complex as you require. The source modules as delivered are referred to as default exits because the logic they contain is executed if no modifications are made. The source for the default exits are in the CA Gen installation directory.

Visual Studio Support

CA Gen supports compiling generated C applications on Windows using Visual Studio 2010.

The %GENxx%Gen\VS100 folder contains a collection of files that support Visual Studio 2010. A set of user exit rebuild procedures is also present in the %GEN*xx*%Gen\VS100 folder and must be used to rebuild any necessary Visual Studio 2010 designated user exits.

CA Gen supports compiling generated C applications on Windows using Visual Studio 2012.

The %GENxx%Gen\VS110 folder contains a collection of files that support Visual Studio 2012. A set of user exit rebuild procedures is also present in the %GEN*xx*%Gen\VS110 folder and must be used to rebuild any necessary Visual Studio 2012 designated user exits.

64-bit Windows Support

CA Gen supports compiling and executing generated C Blockmode and server applications as 64-bit images on Windows using Visual Studio. The %GEN*xx*%Gen\VSabc\amd64 folder contains a collection of files that support 64-bit Windows. A set of user exit rebuild procedures is also available in the %GEN*xx*%Gen\VSabc\amd64 folder and must be used to rebuild any necessary 64-bit designated user exits.

Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

User Exits Collections

CA Gen provides a large number of user exits that span a number of components and are written in several languages. Below is the collection of the various sets of user exits that will be documented within this guide.

Windows User Exits

- Windows C User Exits
  - Blockmode Runtime User Exits
  - GUI Runtime User Exits
  - Client Middleware User Exits
  - Server Runtime User Exits
  - Server Middleware User Exits
  - C Proxy User Exits
- Windows Java User Exits
- Windows .Net User Exits

UNIX/Linux User Exits

- Blockmode Runtime User Exits
- Client Middleware User Exits
- Server Runtime User Exits
- Asynchronous Daemon User Exits
- Server Middleware User Exits
- C Proxy User Exits

z/OS User Exits

- Blockmode Runtime User Exits
- Middleware User Exits
- Server Runtime User Exits

NonStop User Exits

- Blockmode Runtime User Exits
- Server Runtime User Exits

Web Generation User Exits

Web View User Exits

.NET User Exits

Browser User Exits

User Exit General Information

After the desired user exit source file has been located, make a backup copy of the file, then edit the file and save it back to its original name and location.

Important! Do not modify the package name. Doing so will cause the exit to be unrecognized at runtime.

Because the user exit source files are saved to their original locations, it may be useful to save the original and modified sources in a configuration management system to prevent the changes from being lost should a new installation overwrite the source file.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

Note: Throughout this document, steps are provided to execute several Windows user exit rebuild procedures. Before executing these procedures, verify that the required compiler or linker environment variables are initialized. The Microsoft Visual Studio supplies two batch scripts (when working with Visual Studio) that must be run from within the command window to set these variables.

For Visual Studio, these batch scripts may be referenced through the supplied environment variables VSabcCOMNTOOLS and VCINSTALLDIR as follows:

For 32-bit or 64-bit build environments:

"%VSabcCOMNTOOLS%VSVARS32.BAT"

In addition for 64-bit build environments only:

"%VCINSTALLDIR%VCVARSALL.BAT" x64

Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

In the following chapters, each user exit is documented as follows:

User Exit Name and Title

Acts as heading for User Exit

Functional Usage

Displays calling format, including format, and return variable

Source Code

Indicates name of the source code filename

Purpose

Details the purpose of the user exit

Arguments

Lists a table with arguments and descriptions

Return Code

Lists a table with return codes and descriptions

Default Behavior

Describes how the user exit behaves when delivered (unmodified)

Building on Target System

Describes how to rebuild the user exit

Related User Exits

Displays a list of related user exits

Chapter 2: Windows C User Exits

There are several sets of user exits to support the variety of C applications that run on the Windows platform. Batch files are provided to assist in building the runtime user exits for each runtime environment listed in the following sections.

The following table lists the sets of runtime user exits available on the Windows platform.

User Exit Set	Provided As
Blockmode Runtime	AEUXITxxN.DLL, AECDB2xxN.DLL, AECODBxxN.DLL, AECORAxxN.DLL
GUI Runtime	WRExx0N.DLL, STUBDB2N.EXE, STUBODBN.EXE, STUBORAN.EXE
Client Middleware	
Client Manager	CMICXxxN.DLL, CIDExxN.DLL, DECRExxN.DLL, IEFDIRN.DLL, RSCUXxxN.DLL
Communications Bridge	CIDExxN.DLL, DECRExxN.DLL, ECIUXxxN.DLL, RSCUXxxN.DLL, TCPUXxxN.DLL
Common System Utilities (CSU)	CSUVNxxN.DLL
TCP/IP as Transport	TCPCXxxN.DLL
WebSphere MQ as Transport	MQSCXxxN.DLL
ECI as Transport	ECICXxxN.DLL
Tuxedo as Transport	TXWCXxxN.DLL, TXCXxxN.DLL
Web Services as Transport	WSCXxxN.DLL

User Exit Set	Provided As
Server Runtime	AEUEXITxxN.DLL, AECDB2xxN.DLL, AECODBxxN.DLL, AECORAxxN.DLL
Asynchronous Daemon	AEFSECEX.EXE
Server Middleware	
WebSphere MQ as Transport	MQSSXxxN.DLL
C Proxy	PREXxxN.DLL

Windows Blockmode User Exits

The following table summarizes the functions available through the user exits for generated blockmode applications:

Name	Description
DBCOMMIT	Database Commit User Exit. There is one user exit routine for each supported database: ODBC, Oracle, and DB2.
DBCONNCT	Database Connection User Exit. There is one user exit routine for each supported database: ODBC, Oracle, and DB2.
DBDISCNT	Database Disconnect User Exit. There is one user exit routine for each supported database: ODBC, Oracle, and DB2.
TIRDCRYP	Runtime Cooperative Communications Decryption User Exit
TIRDLCT	Dialect User Exit
TIRDRTL	Default Retry Limit User Exit
TIRELOG	Server Error Logging Exit
TIRHELP	Help Interface User Exit
TIRNCRYP	Cooperative Runtime Communications Encryption User Exit
TIRMTQB	Message Table User Exit
TIRSECR	Security Interface User Exit
TIRSECV	Cooperative Runtime Communications Client Security Validation User
TIRSERRX	Server to Server Error Exit
TIRSYSID	System ID User Exit
TIRTERMA	User Termination User Exit

Name	Description
TIRTIAR	Database Error Message User Exit. There is one user exit routine for each supported database: ODBC, Oracle and DB2.
TIRUPDB	MBCS Uppercase Translation User Exit
TIRUPPR	Uppercase Translation User Exit
TIRURTL	Ultimate Retry Limit User Exit
TIRUSRID	User ID User Exit
TIRXLAT	National Language Translation Exit
TIRYYX	Date User Exit

Note: The database user exits DBCONNCT, DBCOMMIT, DBDISCNT, and TIRTIAR are rebuilt into individual DLLs (AECDB2xxN.DLL, AECODBxxN.DLL, AECORAxxN.DLL) using the command procedure %GEN*xx*%Gen\VSabc\mkdbs.bat when using Visual Studio 32-bit and %GEN*xx*%Gen\VSabc\amd64\mkdbs.bat when using Visual Studio 64-bit.

Blockmode runtime user exits are rebuilt into the DLL AEUEXITxxN.DLL using the command procedure %Gen*xx*%Gen\VSabc\mkexits.bat for Visual Studio 32-bit or %Gen*xx*%Gen\VSabc\amd64\mkexits.bat for Visual Studio 64-bit. This is the same DLL that is used with Server applications.

Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

Details for the preceding user exits follow in a separate section for each.

DBCOMMIT—Database Commit Exit

```
void dbcommit (

int rc)
```

Source Code

For Db2: TIRDB2.PPC

For ODBC: TIRODBC.C

For Oracle: TIRORA.PPC

Purpose

Use the Database Commit User Exit to customize the commit logic for a particular database. The default processing of this user exit provides a simple database commit.

There exists a Database Commit User Exit for each supported DBMS: ODBC, Oracle and DB2.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
rc	Input	Type of commit to perform

Return Code

None

Default Behavior

By default, these modules perform a standard database commit statement.

Building on Windows

The Database Connection User Exit is built as part of the dynamic link libraries AECDB2xxN.DLL, AECODBxxN.DLL and AECORAxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates the platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKDBS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

3. Run MKDBS.BAT, passing the DBMS as the parameter. The DBMS is one of the following:
   - DB2
   - ODBC
   - ORACLE
   - ORACLE 10

Note: For more information about building the Blockmode Runtime Database DLLs, see the *Windows Implementation Toolset User Guide*.

Related User Exits

DBCONNCT, DBDISCNT, TIRTIAR

DBCONNCT—Database Connection User Exit

```
int dbconnect (
    char *user,
    char *pswd)
```

Source Code

For Db2: TIRDCONN.SQC

For ODBC: TIRCODBC.C

For Oracle: TIROCONN.PC

Purpose

Use the Database Connection User Exit to customize the connection to the particular database. This user exit enhances database security. The default processing of this user exit provides a simple database connection.

There exists a Database Connect User Exit for each supported DBMS: ODBC, Oracle and DB2.

The default method for CA Gen to acquire DBMS connection information for blockmode applications is for the AEF to locate the trancode in the AEENV file. Connection information includes the username, password, and database name. Each person that executes the application must have read access to the AEENV file that contains the connection information.

Database	Description	Host Variable	Declared Type
Oracle	user ID	uid	VARCHAR(32)
Oracle	Password	pwd	VARCHAR(32)
DB2	user ID	uid	char(9)
DB2	Password	pwd	char(9)
DB2	database name	dbname	char(9)
ODBC	user ID	uid	char(20)
ODBC	Password	pwd	char(20)
ODBC	database name	dbname	char(31)

We recommend leaving the call to dbid() unchanged, and adding logic immediately before the database connect statements to populate the appropriate variables. Ensure that you add all code that the DBMS requires. For example, verify arr and len elements are populated correctly for VARCHAR. We also recommend that all AEENV files contain character strings as place holders for the database connection information. These character strings do not have to contain valid connection information.

For greater security, add a call to an encryption routine.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*user	Input	Pointer to DBMS userid

Name	I/O	Description
*pswd	Input	Pointer to DBMS password for userid

Return Code

Integer representing success or failure of database connection.

Default Behavior

By default, these modules read the database connect information from the AEENV file and use the information in the database connect statement.

Building on Windows

The Database Connection User Exit is built as part of the dynamic link libraries AECDB2xxN.DLL, AECODBxxN.DLL and AECORAxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKDBS.BAT (%GENxx%Gen\VSabc for Visual Studio 32-bit or %GENxx%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKDBS.BAT, passing the DBMS as the parameter. The DBMS is one of the following:

   DB2

   ODBC

   ORACLE

Note: For more information about building the Blockmode Runtime Database DLLs, see the *Windows Implementation Toolset User Guide*.

Related User Exits

DBCOMMIT, DBDISCNT

DBDISCNT—Database Disconnect Exit

```
int dbdiscnt()
```

Source Code

For Db2: TIRDB2.PPC

For ODBC: TIRODBC.C

For Oracle: TIRORA.PPC

Purpose

Use the Database Disconnect User Exits to customize the database disconnect. The default processing of these users exits provides simple database disconnect.

There exists a Database Disconnect User Exit for each supported DBMS: ODBC, Oracle and DB2.

Arguments

None

Return Code

Integer representing success or failure of database disconnection.

Default Behavior

By default, these modules perform a standard database disconnect statement.

Building on Windows

The Database Connection User Exit is built as part of the dynamic link libraries AECDB2xxN.DLL, AECODBxxN.DLL and AECORAxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKDBS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit, where abc is the supported version of Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKDBS.BAT, passing the DBMS as the parameter. The DBMS is one of the following:

   DB2

   ODBC

   ORACLE

Note: For more information about building the Blockmode Runtime Database DLL's, see the *Windows Implementation Toolset User Guide*.

Related User Exits

DBCONNCT, DBCOMMIT

TIRDLCT—Dialect Exit

```
void TIRDLCT (
    char *rp1,
    char *rp2,
    struct dialect_cmcb *tirdlct_cmcb)
```

Source Code

TIRDLCT

Purpose

TIRDLCT supplies the current user's dialect to the application and is useful only for multilingual applications. For multilingual support, the user is responsible for modifying this module to return the appropriate dialect. The dialect returned should be defined using the Design selection on the CA Gen action bar. If it is not, the application's default dialect is used.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*tirdlct_cmcb	Input/Output	A pointer to a structure containing the following items:
tirdlct_userid	Input	An 8-byte character array containing the current user id as provided by TIRUSRID.
tirdlct_terminal_id	Input	An 8-byte character array containing the current terminal id.
tirdlct_system_id	Input	An 8-byte character array containing the current system id as provided by TIRSYSID.
tirdlct_return_dialect	Input	An 8-byte character array containing the returned dialect.

Return Code

None

Default Behavior

TIRDLCT returns a dialect value of DEFAULT.

Building on Windows

The Dialect User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit where abc is the supported version of Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

TIRUSRID, TIRSYSID

TIRDRTL—Default Retry Limit Exits

```
int tirdrtl (
    char retry_flag)
```

Source Code

TIRDRTL.C

Purpose

TIRDRTL lets you override the CA Gen-defined default value for the TRANSACTION RETRY LIMIT system attribute. TRANSACTION RETRY LIMIT will be initialized to this value at the beginning of each new transaction. This value can subsequently be modified by a SET TRANSACTION RETRY LIMIT statement in an action diagram.

TRANSACTION RETRY LIMIT is used to specify the maximum number of times to retry a transaction when one of the following events occurs:

- A RETRY TRANSACTION action diagram statement executes.

- A deadlock or timeout occurs trying to access a database, and there is no WHEN DATABASE DEADLOCK OR TIMEOUT statement for that entity action statement.

In these cases, uncommitted database updates are rolled back, and an attempt is made to execute the application again. After the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit User Exit (see TIRURTL), no more retries can occur, and the application fails with a runtime error.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
retry_flag	Input	Flag to indicate whether or not to set a retry limit.

Return Code

Integer containing the retry limit.

Default Behavior

If the Default Retry Limit User Exit is not used, TRANSACTION RETRY LIMIT will be initialized to 10 for all target environments. If the Default Retry Limit User Exit is used, it must not return a value greater than that specified in the Ultimate Retry Limit User Exit.

Building on Windows

The Default Retry User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.

2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit where abc is the supported version of Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

3. Run MKEXITS.BAT.

Related User Exits

TIRURTL

TIRHELP—Help Interface Exit

```
void TIRHELP (
    char *rp1,
    char *rp2,
    struct tirhelp *in_tirhelp_cmcb,
    char *in_tirhelp_return_message,
    char *in_environment_list,
    char *in_application_list,
    struct scmgr *in_scmgr_cmcb)
```

Source Code

TIRHELP.C

Purpose

TIRHELP is called when a HELP or PROMPT command is entered. From TIRHELP, a help system can be invoked to provide application help information.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.

Name	I/O	Description
*in_tirhelp_cmcb	Input/Output	A pointer to a structure containing the following items:
tirhelp_request_code	Input/Output	A 2-byte character array containing the type of help requested.
tirhelp_return_code	Output	A 2-byte character array containing the return code.
tirhelp_mapname	Input	An 8-byte character array containing the mapname.
tirhelp_data_filler	Unused	An 8-byte character array used as a structure filler.
tirhelp_trancode	Input	An 8-byte character array containing the trancode.
tirhelp_userid	Input	An 8-byte character array containing the user id.
tirhelp_terminal_id	Input	An 8-byte character array containing the terminal id.
tirhelp_printer_id	Input	An 8-byte character array containing the printer id.
tirhelp_dialect	Input	An 8-byte character array containing the dialect.
tirhelp_message_table	Input	An 8-byte character array containing the message table. This value is passed to TIRMTQB.
tirhelp_filler	Unused	A 16-byte character array used as a structure filler.
tirhelp_last_command	Input	An 80-byte character array containing the last command.
tirhelp_last_message	Input	An 80-byte character array containing the last message.
tirhelp_screen_helpid	Output	A 44-byte character array containing the help identifier for the screen.
tirhelp_field_helpid	Output	A 44-byte character array containing the help identifier for the field.
tirhelp_field_token1	Input	A 3-byte character array containing a field token.
tirhelp_field_token2	Input	A 3-byte character array containing a second field token
tirhelp_field_len	Input	A 3-byte character array containing the field length.
tirhelp_field_value	Input	A 256-byte character array containing the value of the field.
tirhelp_field_protect	Input	A single character containing a field protection flag.
tirhelp_field_intens	Input	A single character containing a field intensity flag.

Name	I/O	Description
in_tirhelp_return_message	Output	An 80-byte character array representing the returned help message. By default, this message is returned from a call to TIRMTQB.
in_environment_list	Input	A pointer to an environment control block. Reserved for runtime internal use only.
in_application_list	Input	A pointer to an application control block. Reserved for runtime internal use only.
in_scmgr_cmcb	Input/Output	A pointer to a screen management control block.

Return Code

None

Default Behavior

The TIRHELP routine will return a message indicating no help is available.

Building on Windows

The Help Interface User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates the platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

TIRMTQB

TIRMTQB—Message Table Exit

```
void TIRMTQB(char *rp1,
    char *rp2,
    char *msg_tbl_name,
    short *msgnum,
    struct PARMMSG *prm);
```

Source Code

TIRMTQB.C

Purpose

This message table exit is called by the runtime when a system-level message is to be displayed. The user can customize the wording of the messages within this exit. Additional tables can also be defined to support other dialects.

The default table includes an entry for each CA Gen runtime error message. Each entry includes the following information:

- Message Number—The message number is permanently assigned by CA Gen. Each message has a unique number.
- Message Text—The message text is the actual words that appear on the application screen when an error occurs. The message text, and any variable values that can be appended, is truncated if it exceeds the length of the error message line defined for the application screen. The error message line is a maximum of 80 characters of which 12 are reserved for the message number.

If the message number is not in the table, TIRMTQB returns a default message.

Runtime Error Table

The Runtime Error Message Table includes an entry for each runtime error message. Each table entry includes the following information:

- Message type— a message number is not found in the table, or when you return to a transaction screen after a fatal error or a Dialog Manager error occurs. Valid message types are shown in the following list:
- Default message— a message number is not found in the table, or when you return to a transaction screen after a fatal error or a Dialog Manager error occurs.
- Dialog Manager error—Occurs when the Dialog Manager is unable to communicate with the system. This is a fatal error beyond the control of CA Gen. An error in the load module packaging or in the configuration specifications causes a Dialog Manager error. Error handling is the same as for a fatal error.

- Fatal error— a CA Gen application abnormal program ending. If a condition occurs at runtime that the generated code cannot handle, the system issues a fatal error. An error message screen displays the appropriate error messages.
- Function error—Occurs if a CA Gen-supported function receives invalid input or produces invalid output. CA Gen-supplied functions manipulate characters, numbers, dates, and times.
- Screen edit error—A non-fatal error that occurs when an input or output value for a field does not match the expected value, the range, type, or format defined for the field during model development. This type of message is displayed on your transaction screen. You can correct the error and continue with the transaction.
- Unformatted input error—Occurs when the unformatted input contains invalid parameters, delimiters, or both. Unformatted input is a list of parameters associated with a clear screen transaction code.
- Message number—Each message has a unique number that is permanently assigned by CA Gen.
- Message text—The message text consists of the actual words that appear on the application screen when an error occurs. Because of the length of the message identifier, the message text is limited to 68 characters for an 80-character screen. The message text and appended variables are truncated if they exceed the length of the error message line defined for the application screen.
- Suffix—(If applicable) The suffix contains variable values, such as return codes, permitted values, or the values in error.

Runtime Error Handling

Runtime errors are handled by the Dialog Manager. Runtime errors are non-fatal, such as screen edit, or fatal errors.

If a non-fatal error such as invalid user input occurs, the Dialog Manager displays an error message on the transaction screen. You can correct the error and continue processing the transaction.

If an application fails because of a fatal error, transaction processing terminates, and the error processing is as follows:

- The Dialog Manager performs all necessary rollbacks of the databases.
- CA Gen displays an error message screen that lists the appropriate runtime error messages.
- Pressing Enter from the error message screen causes CA Gen applications to terminate execution.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*msg_tbl_name	Input	A character string containing the name of the table to be used for extraction of the message text. Currently one table named DEFAULT is supported by the CA Gen runtime.
*msgnum	Input	A short value containing the message number corresponding to the text to be fetched.
*prm	Input/ Output	A pointer to a PARMMSG structure to contain the returned message text information. This structure, defined in tirmtq.h, has the following definition:
PARMLEN		A short value containing the total length of PARMNO + PARMTXT.
PARMNO	Output	An 11-byte character array containing the message number formatted in a standard style.
PARMTXT	Output	A string containing the text corresponding to the error message number. The string can be up to 245 bytes, including the terminating NULL.
filler	Unused	Two unused filler characters

Return Code

None

Default Behavior

The table in the default exit is used to retrieve runtime error message text.

Building on Windows

The Message Table User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates the platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

None

TIRSECR—Security Check Interface Exit

```
void TIRSECR(char * rp1,
    char * rp2,
    struct security_cmcb * in_tirsecr_cmcb);
```

Source Code

TIRSECR.C

Purpose

The Dialog Manager calls the Security Check Interface Exit when a transaction is started and before execution of a dialog flow. This allows transaction-level security checking to be implemented. The following data is provided by the dialog manager of each load module for use in checking security authorization:

- System ID (as provided by the System ID Exit, TIRSYSID)
- User ID (as provided by the User ID Exit, TIRUSRID)

- Trancode
- Terminal ID
- Load module name
- Procedure step name

If the user defined security check passes, TIRSECR should move a value of spaces to the return code. If the security check fails, a non-blank value should be moved to the return code with a message describing the violation inserted into the tirsecr_failure_msg buffer. The current dialect in effect on the client is passed in using tirsecr_dialect.

When the dialog manager receives control, it proceeds with the transaction if the return code is spaces, or issues an error if it is not.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*in_tirsecr_cmcb	Input/Output	A pointer to a structure containing the following items:
tirsecr_userid	Input	An 8-byte character array containing the security user ID as provided by the TIRUSRID user exit
tirsecr_trancode	Input	An 8-byte character array containing the current transaction code.
tirsecr_terminal_id	Input	An 8-byte character array containing the current terminal ID.
tirsecr_system_id	Input	An 8-byte character array containing the current system ID as returned by the TIRSYSID user exit.
tirsecr_load_module	Input	An 8-byte character array containing the name of the executing load module calling this exit.
tirsecr_pstep_name	Input	A 32-byte character array containing the name of procedure step being executed.
tirsecr_dialect	Input	A 32-byte character array containing the dialect in effect on the client.

Name	I/O	Description
tirsecr_return_code	Output	A 2-character array representing the success or failure of this exit processing. TIRSECR_ALL_OK defined as two spaces implies success, any other value implies failure. If none spaces are return, tirfail will be passed the tirsecr_failure_msg message.
tirsecr_failure_msg	Output	An 80-byte character array used in conjunction with a failing return code in tirsecr_return_code. This exit can insert an error message into this array that will be passed by the Dialog manager to the tirfail user exit.

Return Code

None directly. For more information, see tirsecr_return_code structure member.

Default Behavior

The default exit will return a status code of spaces, indicating no security violation was detected.

Building on Windows

The Security Check User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates the platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

TIRUSRID, TIRSYSID

TIRSYSID—System ID Exit

```
void TIRSYSID ( char *rp1;
    char *rp2;
    char *system_id);
```

Source Code

TIRSYSID.C

Purpose

TIRSYSID supplies the system ID to the application.

The purpose of TIRSYSID is to implement application logic that lets you implement one model on multiple platforms, and perform processing appropriate for the platform. The system ID is also one of the parameters passed to the Security Interface Exit (TIRSECR).

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*system_id	Output	An 8-byte character array representing the system identifier where the server application is executing.

Return Code

None

Default Behavior

By default, TIRSYSID calls the runtime routine DEFSYSID. This routine returns a default system ID, the value of which depends on the platform on which the application is executing.

Under Windows if the environment variable IEF_SYSID is set, the first 8 characters of this variable are used. Otherwise, "WIN" is returned.

Building on Windows

The System ID User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

TIRSECR

TIRTERMA—User Termination Exit

```
void TERTERMA (
    char *rp1,
    char *rp2,
    struct term_pb *pb)
```

Source Code

TIRTERMA.C

Purpose

TIRTERMA is called when an application fails. Modification of TIRTERMA lets the user customize the handling of runtime errors.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*pb	Input/ Output	A pointer to a PARMMSG structure to contain the termination information. This structure is defined in tirterma.h.

Return Code

None

Default Behavior

The default processing for TIRTERMA returns a status code of spaces, indicating to use standard error handling.

Building on Windows

The User Termination User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

None

TIRTIAR—Database Error Message Exit

```
void TIRTIAR (
    char *sqlca,
    short *ml,
    char *mb,
    int maxLength)
```

Source Code

For Db2: TIRDB2.PPC

For ODBC: TIRODBC.C

For Oracle: TIRORA.PPC

Purpose

Use the Database Error Message User Exit to customize the error message received from the database commit. The default processing of this user exit provides a simple database error message.

There exists a Database Error Message User Exit for each supported DBMS: ODBC, Oracle and DB2.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*sqlca	Input	Pointer to SQLCA structure
*ml	Output	Pointer to error message length
*mb	Output	Pointer to error message buffer
maxLength	Input	Maximum length of error message that can be written to error message buffer

Return Code

None

Default Behavior

By default, these modules provide the error message returned by the database commit.

Building on Windows

The Database Connection User Exit is built as part of the dynamic link libraries AECDB2xxN.DLL, AECODBxxN.DLL and AECORAxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKDBS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKDBS.BAT, passing the DBMS as the parameter. The DBMS is one of the following:

   DB2

   ODBC

   ORACLE

Note: For more information about building the Blockmode Runtime Database DLLs, see the *Windows Implementation Toolset User Guide.*

Related User Exits

DBCOMMIT

TIRUPDB—MBCS Uppercase Translation Exit

```
void TIRUPDB (
    char *rp1,
    char *rp2,
    char *tbl_name,
    long *len,
    char *xlate_data)
```

Source Code

TIRUPDB.C

Purpose

TIRUPDB is called to uppercase multi-byte text. The user can modify the mechanism used to uppercase multi-byte text with this user exit.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*tbl_name	Input	A pointer to a translation table name.
*len	Input/Output	Length of text to convert to uppercase.
*xlate_data	Input/Output	A pointer to the text to be uppercased.

Return Code

None

Default Behavior

The default translation uses MBCS functions to perform uppercase translation based upon the active system code page. However, the system designer, programmer, may add code to recognize dialects and perform any lower to upper functionality desired. In that case, insure that the default behavior still uses the MBCS libraries.

Building on Windows

The MBCS Uppercase Translation User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

TIRUPPR

TIRUPPR—Uppercase Translation Exit

```
void TIRUPPR (
    char *rp1,
    char *rp2,
    char *tbl_name,
    long *len,
    char *xlate_data)
```

Source Code

TIRUPPR.C

Purpose

TIRUPPR is called to uppercase multi-byte text. The user can modify the mechanism used to uppercase multi-byte text with this user exit.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*tbl_name	Input	A pointer to a translation table name.
*len	Input/Output	Length of text to convert to uppercase.
*xlate_data	Input/Output	A pointer to the text to be uppercased.

Return Code

None

Default Behavior

The default translation uses MBCS functions to perform uppercase translation based upon the active system code page. However, the system designer, programmer, may add code to recognize dialects and perform any lower to upper functionality desired. In that case, insure that the default behavior still uses the MBCS libraries.

Building on Windows

The Uppercase Translation User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.

2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

3. Run MKEXITS.BAT.

Related User Exits

TIRUPDB

TIRURTL—Ultimate Retry Limit Exit

```
long tirurtl ()
```

Source Code

TIRURTL.C

Purpose

TIRURTL lets you specify a maximum value for the TRANSACTION RETRY LIMIT system attribute. This value can never be exceeded by a SET TRANSACTION RETRY LIMIT statement in an action diagram, or by the Default Retry Limit User Exit.

After the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit User Exit, no more retries can occur, and the application fails with a runtime error.

Arguments

None

Return Code

Long containing the retry limit.

Default Behavior

If the Ultimate Retry Limit User Exit is not used, the maximum value of TRANSACTION RETRY LIMIT will be 99 for all target environments. The Ultimate Retry Limit User Exit can be modified to return a value of zero to suppress all retry attempts.

Building on Windows

The Ultimate Retry User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates the platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

TIRDRTL

TIRUSRID—User ID Exit

```
void TIRUSRID ( char *rp1;
    char *rp2;
    char *filler_parm;
    char *user_id);
```

Source Code

TIRUSRID.C

Purpose

TIRUSRID is used to supply the user's ID to the application. The user ID is one of the parameters passed to the Security Interface Exit (TIRSECR).

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*filler_parm	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*user_id	Output	A pointer to an 8-byte character array into which the user ID can be returned.

Return Code

None

Default Behavior

The default action taken by this module is to call runtime routine DEFUSRID which returns a default user ID, the value of which depends on the platform on which the system is executing.

Building on Windows

The User ID User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

TIRSECR

TIRYYX—Date Exit

```
void TIRYYX (
    struct tiryyx_param_block *pb)
```

Source Code

TIRYYX.C

Purpose

TIRYYX is used to process two-digit or yy-style date input and to set the century part using any fixed-window, sliding-window, or other algorithm of choice, when using CA Gen in the standard map generation mode.

Internally, CA Gen handles four digit year dates correctly assuming the user application uses the yyyy edit pattern throughout. If the user interface is designed to accept a two-digit date entry, and defaulting to the current century is not acceptable, use this exit to implement logic to get the required behavior for defaulting the century part of the date.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*pb	Input/Output	A pointer to a tiryyx structure containing the following items:
return_code	Output	A 4-byte character array containing the current year
current_year	Input	A 4-byte character array containing the current year.
edit_year	Input/Output	A 4-byte character array containing the edit year.

Return Code

None

Default Behavior

The default user exit behavior does not perform any processing and returns.

Building on Windows

The Date User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

None

Windows GUI Client User Exits

The following table summarizes the functions available through the GUI runtime user exits:

Name	User Exit Description
C4COMMIT	Database Commit User Exit. There is one user exit routine for each supported database: ODBC, Oracle and DB2.
C4CONNECT	Database Connect User Exit. There is one user exit routine for each supported database: ODBC, Oracle and DB2.
C4DISCONNECT	Database Disconnect User Exit. There is one user exit routine for each supported database: ODBC, Oracle and DB2.

Name	User Exit Description
C4ERRMSG	Database Message User Exit. There is one user exit routine for each supported database: ODBC, Oracle and DB2.
C4ROLLBACK	Database Rollback User Exit. There is one user exit routine for each supported database: ODBC, Oracle and DB2.
C4SQLLEN	Database SQLCA Len User Exit. There is one user exit routine for each supported database: ODBC, Oracle, and DB2.
WRASYNCSRVRERROR	Client/Server Asynchronous Flow Server Failure Exit
WRDEFAULTYEAR	Century Default Exit
WRDRTL	Default Retry Limit Exit
WRGLB	Globalization Exit
WRSECDECRYPT	Client/Server Decryption Exit
WRSECENCRYPT	Client/Server Encryption Exit
WRSECTOKEN	Client Security Token User Exit
WRSRVRERROR	Client/Server Flow Failure Exit
WRSTRNCM	String Comparison User Exit
WRSYSID	System ID Exit
WRTERMID	Terminal ID Exit
WRUPPR	Uppercase Translation Exit
WRURTL	Ultimate Retry Limit Exit
WRUSRID	User ID Exit

Note: The database user exits C4CONNECT, C4DISCONNECT, C4COMMIT, C4ROLLBACK, C4ERRMSG, and C4SQLLEN are rebuilt into individual stub executables using the make procedures stubdb2n.mak (Db2), stuboran.mak (Oracle), and stubodbn.mak (ODBC) found in %GENxx%Gen\VSabc for Visual Studio.

GUI runtime user exits are rebuilt into the DLL WRExx0N.DLL using the command procedure %GENxx%Gen\VSabc\mkexitsn.bat for Visual Studio.

Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

Details for the preceding user exits follow in a separate section for each.

C4COMMIT—Database Commit Exit (Windows)

```
int C4COMMIT (
    char *buff,
    char *gblSqlca)
```

Source Code

For Db2: STUBDB2N.SQC

For ODBC: STUBODBN.C

For Oracle: STUBORAN.SQC

Purpose

Use the Database Commit User Exit to customize the database commit.

There exists a Database Commit User Exit for each supported DBMS: ODBC, Oracle and DB2.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*buff	Output	Pointer to buffer that contains any error message that was obtained while performing the database commit.
*glbSqlca	Output	Pointer to SQLCA structure.

Return Code

Integer representing success or failure of database commit.

Default Behavior

The default processing of this user exit provides a simple database commit.

Building on Windows

After modifying the user exit, use the make procedure stubdb2n.mak (Db2), stuboran.mak (Oracle), or stubodbn.mak (ODBC) in %GENxx%Gen\VSabc for Visual Studio to precompile, compile, and link the stub executable so that all applications have access to the new logic.

Note: xx refers to the current release of CA Gen and abc refers to the supported version of Visual Studio. For the current release number and the the supported version of Visual Studio, see the Release Notes.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run nmake –nologo –s –f stub*dbms*n.mak all

   where *dbms* is either ‘db2’, ‘ora’ or ‘odb’

Additional qualifiers may be available for each DBMS. See the comments section of each .mak file.

Related User Exits

C4CONNECT, C4DISCONNECT, C4ROLLBACK, C4ERRMSG, C4SQLLEN

C4CONNECT—Database Connection Exit (Windows)

```
int C4CONNECT (
    char *szDatabase,
    char *buff,
    char *gblSqlca,
    HINSTANCE hResDll)
```

Source Code

For Db2: STUBDB2N.SQC

For ODBC: STUBODBN.C

For Oracle: STUBORAN.SQC

Purpose

Use the Database Connection User Exit to customize the connection to the particular database. This user exit enhances database security. The default processing of this user exit provides a simple database connection.

There exists a Database Connect User Exit for each supported DBMS: ODBC, Oracle and DB2.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*szDatabase	Input	Pointer to a database structure containing the userid, password and database name.
*buff	Output	Pointer to buffer that contains any error message that was obtained while performing the database connection.
*glbSqlca	Output	Pointer to SQLCA structure.
hResDll	Input	Pointer to Resource DLL in order to obtain error messages.

Return Code

Integer representing success or failure of database connection.

Default Behavior

The default action here is for the connect function to call into the existing GDIC function to get the current connect information. It is possible to have the user change the database name during the GDIC function so the database name should be checked after the return from GDIC.

Building on Windows

After modifying the user exit, use the make procedure stubdb2n.mak (Db2), stuboran.mak (Oracle), or stubodbn.mak (ODBC) in %GENxx%Gen\VSabc for Visual Studio to precompile, compile, and link the stub executable so that all applications have access to the new logic.

Note: xx refers to the current release of CA Gen and abc refers to the supported version of Visual Studio. For the current release number and the the supported version of Visual Studio, see the Release Notes.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run nmake –nologo –s –f stub*dbms*n.mak all

   where *dbms* is either ‘db2’, ‘ora’ or ‘odb’

Additional qualifiers may be available for each DBMS. See the comments section of each .mak file.

Related User Exit

C4DISCONNECT, C4COMMIT, C4ROLLBACK, C4ERRMSG, C4SQLLEN

C4DISCONNECT—Database Disconnect Exit (Windows)

```
int C4DISCONNECT (
    char *name,
    char *buff,
    char *gblSqlca)
```

Source Code

For Db2: STUBDB2N.SQC

For ODBC: STUBODBN.C

For Oracle: STUBORAN.SQC

Purpose

Use the Database Disconnection User Exit to customize the database disconnect.

There exists a Database Connect User Exit for each supported DBMS: ODBC, Oracle and DB2.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*name	Input	Pointer to the database name.
*buff	Output	Pointer to buffer that contains any error message that was obtained while performing the database disconnect.
*glbSqlca	Output	Pointer to SQLCA structure.

Return Code

Integer representing success or failure of database disconnection.

Default Behavior

The default processing of this user exit provides a simple database disconnection.

Building on Windows

After modifying the user exit, use the make procedure stubdb2n.mak (Db2), stuboran.mak (Oracle), or stubodbn.mak (ODBC) in %GENxx%Gen\VSabc for Visual Studio to precompile, compile, and link the stub executable so that all applications have access to the new logic.

Note: xx refers to the current release of CA Gen and abc refers to the supported version of Visual Studio. For the current release number and the the supported version of Visual Studio, see the Release Notes.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run nmake –nologo –s –f stub*dbms*n.mak all

   where *dbms* is either ‘db2’, ‘ora’ or ‘odb’

Additional qualifiers may be available for each DBMS. See the comments section of each .mak file.

Related User Exits

C4CONNECT, C4COMMIT, C4ROLLBACK, C4ERRMSG, C4SQLLEN

C4ERRMSG—Database Message Exit (Windows)

```
int C4ERRMSG (
    char *inmsg,
    char *outmsg,
    char *sblSqlca)
```

Source Code

For Db2: STUBDB2N.SQC

For ODBC: STUBODBN.C

For Oracle: STUBORAN.SQC

Purpose

Use the Database Message User Exit to customize the database error messaging.

There exists a Database Message User Exit for each supported DBMS: ODBC, Oracle and DB2.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*inmsg	Input	Pointer to incoming error message.
*outmsg	Output	Pointer to outgoing error message.
*glbSqlca	Input	Pointer to SQLCA structure.

Return Code

Integer representing success or failure of database error retrieval.

Default Behavior

The default processing of this user exit provides a simple database error message management.

Building on Windows

After modifying the user exit, use the make procedure stubdb2n.mak (Db2), stuboran.mak (Oracle), or stubodbn.mak (ODBC) in %GENxx%Gen\VSabc for Visual Studio to precompile, compile, and link the stub executable so that all applications have access to the new logic.

Note: xx refers to the current release of CA Gen and abc refers to the supported version of Visual Studio. For the current release number and the the supported version of Visual Studio, see the Release Notes.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run nmake –nologo –s –f stub*dbms*n.mak all

   where *dbms* is either ‘db2’, ‘ora’ or ‘odb’

Additional qualifiers may be available for each DBMS. See the comments section of each .mak file.

Related User Exits

C4CONNECT, C4DISCONNECT, C4COMMIT, C4ROLLBACK, C4SQLLEN

C4ROLLBACK—Database Rollback Exit (Windows)

```
int C4ROLLBACK (
    char *buff,
    char *gblSqlca)
```

Source Code

For Db2: STUBDB2N.SQC

For ODBC: STUBODBN.C

For Oracle: STUBORAN.SQC

Purpose

Use the Database Rollback User Exit to customize the database rollback.

There exists a Database Rollback User Exit for each supported DBMS: ODBC, Oracle and DB2.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*buff	Output	Pointer to buffer that contains any error message that was obtained while performing the database rollback.
*glbSqlca	Output	Pointer to SQLCA structure.

Return Code

Integer representing success or failure of database rollback.

Default Behavior

The default processing of this user exit provides a simple database rollback.

Building on Windows

After modifying the user exit, use the make procedure stubdb2n.mak (Db2), stuboran.mak (Oracle), or stubodbn.mak (ODBC) in %GENxx%Gen\VSabc for Visual Studio to precompile, compile, and link the stub executable so that all applications have access to the new logic.

Note: xx refers to the current release of CA Gen and abc refers to the supported version of Visual Studio. For the current release number and the the supported version of Visual Studio, see the *Release Notes*.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

3. Set Microsoft Visual Studio compiler environment variables.
4. Run nmake –nologo –s –f stub*dbms*n.mak all

   where *dbms* is either ‘db2’, ‘ora’ or ‘odb’

Additional qualifiers may be available for each DBMS. See the comments section of each .mak file.

Related User Exits

C4CONNECT, C4DISCONNECT, C4COMMIT, C4ERRMSG, C4SQLLEN

C4SQLLEN—Database SQLCA Length Exit (Windows)

```
int C4SQLLEN ()
```

Source Code

For Db2: STUBDB2N.SQC

For ODBC: STUBODBN.C

For Oracle: STUBORAN.SQC

Purpose

Use the Database SQLCA Length User Exit to customize the database sqlca length.

There exists a Database SQLCA Length User Exit for each supported DBMS: ODBC, Oracle and DB2.

Arguments

None

Return Code

Integer representing success or failure of database SQLCA length retrieval.

Default Behavior

The default processing of this user exit provides the length of the SQLCA structure of the current DBMS.

Building on Windows

After modifying the user exit, use the make procedure stubdb2n.mak (Db2), stuboran.mak (Oracle), or stubodbn.mak (ODBC) in %GENxx%Gen\VSabc for Visual Studio to precompile, compile, and link the stub executable so that all applications have access to the new logic.

Note: xx refers to the current release of CA Gen and abc refers to the supported version of Visual Studio. For the current release number and the the supported version of Visual Studio, see the *Release Notes*.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run nmake –nologo –s –f stub*dbms*n.mak all

   where *dbms* is either ‘db2’, ‘ora’ or ‘odb’

Additional qualifiers may be available for each DBMS. See the comments section of each .mak file.

Related User Exit

C4CONNECT, C4DISCONNECT, C4COMMIT, C4ROLLBACK, C4ERRMSG

WRASYNCSRVERROR—Asynchronous Flow Server Failure Exit (Windows)

```
int WRASYNCSRVRERROR ( char statementType,
    long requestId,
    char *viewLabel,
    char responseScope,
    int failureType,
    char *failureCommand,
    ErrorList errorList,
    ErrorToken errorToken);
```

Source Code

WREXITN.C

Purpose

This user exit is invoked when an error is detected during the processing of an asynchronous client to server flow. The parameter failureType describes the origin of this error.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
statementType	Input	This character represents which of the supported asynchronous action statements was being processed at the time of the flow failure. Valid Values are: E: USE ASYNC: Response Type Notify P: USE ASYNC: Response Type Poll N: USE ASYNC: Response Type No Response G: GET ASYNC RESPONSE: non blocking, WHEN PENDING clause has not been coded B: GET ASYNC RESPONSE: blocking, WHEN PENDING clause has been coded C: CHECK ASYNC RESPONSE I: IGNORE ASYNC RESPONSE
requestId	Input	A long value containing a unique request identifier contained within the ASYNC REQUEST View corresponding to the IDENTIFIED BY clause of the statement currently being executed. This value is returned from the supporting runtime using the flow initiating USE ASYNC statement execution.
*viewLabel	Input	A character pointer to the label attribute of the CA Gen action diagram statement has associated Async Request View.
responseScope	Input	The response scope as defined within a USE ASYNC statement. Only valid if the current action diagram statement is a USE ASYNC. Valid values are: "G"—the response scope is global to the client executable. "P"—the response scope is limited to the executing procedure step.

Name	I/O	Description
failureType	Input	An integer value describing the source of the failure. The value can be one of the following: "CFBUILD" is an error in the construction or parsing of a client/server flow message or response. "XFAL" identifies an error during the server procedures action block execution. "XERR" identifies a communications error occurring somewhere between construction of a message or response, and the deciphering of that message by the partner in this flow. "XASY," if a USE ASYNC statement is being processed, identifies a communications error occurring somewhere between construction of a message or response, and the deciphering of that message by the partner in this flow. If a GET ASYNC RESPONSE, CHECK ASYNC RESPONSE, or IGNORE ASYNC RESPONSE is being processed, this fail type indicates the Request Id within the view identified by the "viewLabel" is invalid.
*failureCommand	Output	A character array, containing a maximum of 9 characters including the terminating NULL. This array can be populated with a command used to reinvoke the failing procedure step. This command is only used when returning from the exit with a FailAction of "serverFailedRestart." It will be ignored for any other FailAction.
errorList	Input/Output	An array of characters representing message strings constructed by and normally displayed using the ErrorReport dialog to describe the failure. Each message string is null terminated. Newline characters for formatting are also present as required. The complete list is terminated by more-than-one contiguous null character. On a "serverFailedDisplay" return, errorList, as modified in this exit, will be displayed in the ErrorReport dialog. errorList has a maximum length of 2048 bytes.

Name	I/O	Description
errorToken	Input	An array of characters, errorToken is only used with XFAL messages. errorToken can contain a token constructed by the Error Logging exit (TIRELOG) linked with the server load module describing the server action block failure. errorToken has a maximum length of 4097 bytes.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
serverFailedDisplay	This return value does not let the standard error report display if an appropriate WHEN clause has been coded for the executing action diagram statement. Execution will return to the calling procedure step.
serverFailedRestart	This return value suppresses the standard error report and reinvokes the client procedure step that originated the dialog flow. In the case of a failing "procedure step usage," the parent procedure step is returned to at the statement immediately following the Use. In both cases, if failureCommand is set it is used as the system command when reinvoking or returning to the client procedure step. For flows designed to return the server's exit state to the client, the exit state set when reinvoking or returning to the client procedure step will be the last value set by the client.
serverFailedTerminate	This return value will suppress the standard error report dialog, will not attempt to return to the client procedure step, and will redisplay the client window.

Default Behavior

The default return value of “serverFailedDisplay” does not let the standard error report display if an appropriate WHEN clause has been coded for the executing action diagram statement.

Builidng on Windows

The Asynchronous Server Flow Error User Exit is built as part of the dynamic link library WRExx0N.DLL

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITSN.BAT.

Related User Exits

The following are related user exits:

- TIRELOG
- WRSRVRERROR

WRDEFAULTYEAR—Century Default Exit (Windows)

```
int WRDEFAULTYEAR (
    int inYear,
    int current Year)
```

Source Code

WREXITN.C

Purpose

WRDEFAULTYEAR() is invoked when a date or timestamp field receives input and the field's edit pattern specifies a 2-character year value. The current year and the input year are passed to WRDEFAULTYEAR().

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
inYear	Input	Integer containing the 2 or 4 digit year.
currentYear	Input	Integer containing the current year.

Return Code

Integer containing the 4 digit year.

Default Behavior

By default, the current hundred-year value is added to the 2-character year value and returned.

Building on Windows

The Century Default User Exit is built as part of the dynamic link library WRExx0N.DLL, where xx is the CA Gen release number and N indicates platform. A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITSN.BAT.

Related User Exits

TIRYYX

WRDRTL—Default Retry Limit Exit (Windows)

long WRDRTL ()

Source Code

WREXITN.C

Purpose

WRDRTL lets users override the CA Gen-defined default value for the TRANSACTION RETRY LIMIT system attribute. TRANSACTION RETRY LIMIT is initialized to this value at the beginning of each new transaction. A SET TRANSACTION RETRY LIMIT statement in an action diagram might modify this value.

TRANSACTION RETRY LIMIT specifies the maximum number of times to retry a transaction when one of the following events occurs:

- A RETRY TRANSACTION action diagram statement executes.
- A deadlock or timeout occurs trying to access a database, and there is no WHEN DATABASE DEADLOCK OR TIMEOUT statement for that entity action statement.

In these cases, uncommitted database updates are rolled back, and an attempt is made to execute the application again. After the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit , the application fails with a runtime error.

Arguments

None

Return Code

Long containing the retry limit.

Default Behavior

If the Default Retry Limit User Exit is not used, TRANSACTION RETRY LIMIT will be initialized to 10 for all target environments. If the Default Retry Limit User Exit is used, it must not return a value greater than that specified in the Ultimate Retry Limit User Exit.

Building on Windows

The Default Retry Limit User Exit is built as part of the dynamic link library WRExx0N.DLL, where xx is the CA Gen release number and N indicates platform. A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.

2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

3. Run MKEXITSN.BAT.

Related User Exits

WRURTL, TIRDRTL

WRGLB—GIobalization Exit (Windows)

```
int WRGLB (
    char *rp1,
    char *rp2,
    char *separator,
    char *decimal,
    char *currency,
    char *datesep,
    char *timesep,
    char *dateorder)
```

Source Code

WREXITN.C

Purpose

WRGLB supplies runtime information to the GUI, such as the numeric thousands separator character, the numeric decimal point character, the currency symbol, the date separator character, and the date order.

The numeric characters are used when editing numeric fields for output. The characters should correspond to the edit pattern in the model. For example, if the edit pattern in "@ZZZ.ZZZ,99", the WRGLB exit should specify "@" as the currency symbol, "." as the thousands separator, and "," as the decimal point.

Date and time information is used only for date and time fields where the model does not specify the edit pattern. In these cases, the GUI runtime uses this information to build a default edit pattern using the information provided by the WRGLB exit. For example, if the WRGLB exit specifies the date separator as "-" (a dash) and the date order is yymmdd, then the default date edit pattern is yy-mm-dd.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*separator	Input/Output	A pointer to the thousands separator symbol.
*decimal	Input/Output	A pointer to the decimal symbol.
*currency	Input/Output	A pointer to the currency symbol.
*datesep	Input/Output	A pointer to the date separator symbol.
*timesep	Input/Output	A pointer to the time separator symbol.
*dateorder	Input/Output	A pointer to the date order

Return Code

Integer representing the success (1) or failure (0) of the call to WRGLB.

Default Behavior

By default, WRGLB returns the thousands separator, decimal point, date separator, time separator and date order passed in but always copy the Dollar sign to the currency.

Building on Windows

The Globalization User Exit is built as part of the dynamic link library WRExx0N.DLL, where xx is the CA Gen release number and N indicates platform. A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITSN.BAT.

Related User Exits

None

WRSECDECRYPT—Client Decryption Exit (Windows)

```
int WRSECDECRYPT (long maxViewLen,
    long *encryptViewLen,
    unsigned char *encryptView,
    char *failureMsg)
```

Source Code

WREXITN.C

Purpose

The Client Side Decryption exit is called by the GUI runtime when an encrypted response buffer is received from a target server.

The user provides a decryption algorithm that manipulates the data pointed to by encryptView. The encryptViewLen, on input contains the number of bytes available into which the encrypted buffer area can be decrypted. The process of decryption cannot result in a decrypted buffer area that exceeds maxViewLen. If decryption is performed by this exit, EncryptViewLen must be updated with the length of the decrypted result.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
maxViewLen	Input	A long field that contains the maximum available buffer space (in bytes) that the decrypted data can occupy.
*encryptViewLen	Input/Output	On input, EncryptViewLen is the current buffer space (in bytes) of the encrypted data. On output, EncryptViewLen should be updated to contain the length of the decrypted data. The length of the decrypted result cannot exceed maxViewLen.
*encryptView	Input/Output	A pointer to the starting location of the data eligible for being decrypted. The decrypted data must be copied back into this same memory location.

Name	I/O	Description
*failureMsg	Input/Output	The pointer to an 80-character array that can receive a user provided null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is displayed by the GUI runtime.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
DecryptionNotUsed	Indicates to the runtime that the user exit did not perform any decryption of the encrypted data buffer.
DecryptionUsed	Indicates to the runtime that the user exit successfully performed the decryption of the provided encrypted data.
DecryptionFailure	Indicates to the runtime that an error was encountered by the user exit and that the decryption processing has failed The error indication and message string returned using the failureMsg argument will be returned to the GUI Runtime. The GUI runtime will pop up an error message display indicating the failed request.

Default Behavior

The WRSECDECRYPT user exit, as delivered with CA Gen, will return DecryptionNotUsed.

Building on Windows

The Client Decryption User Exit is built as part of the dynamic link library WRExx0N.DLL, where xx is the CA Gen release number and N indicates platform. A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITSN.BAT.

Related User Exits

The following are related user exits:

- TIRDCRYP
- WRSECENCRYPT

WRSECENCRYPT—Client Side Encryption Exit (Windows)

```
int WRSECENCRYPT (char *trancode,
    char *nextLocation,
    char *clientUserid,
    long maxViewLen,
    long *encryptViewLen,
    unsigned char *encryptView,
    char *failureMsg)
```

Source Code

WREXITN.C

Purpose

The Client Side Encryption exit is called by the GUI runtime to provide the opportunity to encrypt a cooperative flow request from Window Manager applications. The data in the Common Format Buffer (CFB) that is eligible to be encrypted include the cooperative flow's view data and optional security offset area.

The user provides an encryption algorithm that consists of manipulating the data pointed to by encryptView. The encryptViewLen, on input contains the number of bytes eligible for being encrypted. The process of encryption cannot result in an encrypted buffer area that exceeds maxViewLen. If encryption is performed by this exit, EncryptViewLen must be updated with the length of the encrypted result. Additionally, this exit must return the EncryptionUsed return code value.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*trancode	Input	A pointer to a character array that contains the trancode associated with the synchronous or asynchronous cooperative flow being processed by the GUI runtime.

Name	I/O	Description
*nextLocation	Input	A pointer to a character array that contains the Next Location associated with the synchronous or asynchronous cooperative flow being processed by the GUI runtime.
*clientUserid	Input	A pointer to a character array that contains the value of the of the CLIENT_USER_ID variable associated with the flow being processed by the GUI runtime synchronous or asynchronous cooperative flow processing. The CLIENT_USESRID variable is optionally set by Action Language coded within the GUI client generated code.
MaxViewLen	Input	A long field that contains the maximum available buffer space (in bytes) that the encrypted data can occupy.
*encryptViewLen	Input/Output	A pointer to a long field. On input, EncryptViewLen is the length of the current buffer space (in bytes) of the data eligible for being encrypted. On output, EncryptViewLen should be updated to contain the length of the encrypted data. The length of the encrypted result cannot exceed maxViewLen.
*encryptView	Input/Output	A character pointer to the starting location of the data eligible for being encrypted. The encrypted data must be copied to this same memory location.
*failureMsg	Input/Output	The pointer to an 80-character array that can receive a user provided null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is displayed by the GUI runtime.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
EncryptionNotUsed	Indicates to the runtime that the user exit did not perform any encryption to the provided data buffer.
EncryptionUsed	Indicates to the runtime that the user exit did perform encryption on the provided data. The runtime marks the CFB as being encrypted. An encrypted CFB will trigger the decryption counterpart user exit to be invoked by the target server manager. The server side decryption user exit is TIRDCRYP.

Return Code	Description
EncryptionFailure	Indicates to the runtime that an error was encountered by the user exit and that the processing of the associated request has failed. The error indication and message string returned using the failureMsg argument would be returned to the GUI runtime. The GUI runtime will pop up an error message display indicating the failed request.

Default Behavior

The WRSECENCRYPT user exit, as delivered with CA Gen, will return EncryptionNotUsed.

Building on Windows

The Client Encryption User Exit is built as part of the dynamic link library WRExx0N.DLL, where xx is the CA Gen release number and N indicates platform. A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Note: xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITSN.BAT.

Related User Exits

The following are related user exits:

- TIRNCRYP
- WRSECDECRYPT

WRSECTOKEN—Client Security Token Exit (Windows)

```
int WRSECTOKEN (char *clientUserid,
    char *clientPassword,
    char *trancode,
    char *nextLocation,
    BOOL *bClntMgrSecurity,
    long *tokenLen,
    char *token,
    char *failureMsg)
```

Source Code

WREXITN.C

Purpose

The Client Side Security Exit is invoked by the GUI runtime to let a user influence how client security data is processed by the GUI runtime code involved in servicing a cooperative flow. Specifically, this exit influences if the Common Format Buffer (CFB) request will contain a security offset and if that data populated in the security offset should be used by other runtime components such as the Client Manager or Communications Bridge when servicing the cooperative flow request.

The trancode and nextLocation variables are provided as input. These input values can be used by the user exit code to determine what return code value should be specified.

In addition to the return code value, this exit has the option of returning some fields as output data to the calling runtime code.

For more information about the input and output fields of this exit routine, see Arguments. For a description on what the invoking GUI runtime will do because of receiving one of the expected return values, see Return Codes.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*clientUserid	Input/Output	A pointer to a character array that contains the value of the of the CLIENT_USER_ID variable associated with the flow being processed by the GUI runtime synchronous or asynchronous cooperative flow processing. The CLIENT_USER_ID variable is optionally set by Action Language coded within the GUI client generated code. The data area pointed to by this argument can be modified by this user exit. The value placed into the referenced data area cannot exceed 64 bytes.
*clientPassword	Input/Output	A pointer to a character array that contains the value of the of the CLIENT_PASSWORD variable associated with the flow being processed by the GUI runtime synchronous or asynchronous cooperative flow operation. The CLIENT_PASSWORD variable is optionally set by Action Language coded within the GUI client generated code. The data area pointed to by this argument can be modified by this user exit. The value placed into the referenced data area cannot exceed 64 bytes.
*trancode	Input	A pointer to a character array that contains the trancode associated with the flow being processed by the GUI runtime synchronous or asynchronous cooperative flow operation.
*nextLocation	Input	A pointer to a character array that contains the Next Location variable associated with the flow being processed by the GUI runtime synchronous or asynchronous cooperative flow operation.
*bClntMgrSecurity	Output	A pointer to an integer Boolean field that can be set to either TRUE or FALSE. The value of this field only has meaning if this user exit returns SecurityUsedEnhanced. TRUE indicates that the security data (Client User ID and Client Password) that is added to the security offset of the associated CFB should be used as the source of the UserID and Password by the Client Manager or Communications Bridge.

Name	I/O	Description
*tokenLen	Input/Output	On input, tokenLen is a pointer to a long integer field that contains the maximum length of the allocated token character buffer. The maximum token length is dependent on the available space remaining during the construction of the CFB. On return from the exit, the long integer pointed to by tokenLen should contain the actual length of data returned in the character array, which is pointed to by the token argument. **Note:** The use of a token is optional, and therefore, setting the long integer pointed to by tokenLen to zero indicates that a token is not specified by the user exit. The length value returned by this field only has meaning if this user exit returns SecurityUsedEnhanced.
*token	Input/Output	On input, token is a pointer to a character array that will accept a user specified security token. The use of a user specified security token is optional. The token data that is provided by this user exit will be provided to the server side TIRSECV security user exit. The security token returned by this field only has meaning if this user exit returns SecurityUsedEnhanced.
*failureMsg	Input/Output	The pointer to an 80-character array that can receive a user provided null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is displayed by the GUI runtime.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
SecurityNotUsed	Indicates to the runtime that the CLIENT_USER_ID, CLIENT_PASSWORD, and security token will NOT be used to populate any part of the cooperative flow request. The client side security variables will not be added to the CFB by the GUI runtime.

Return Code	Description
SecurityUsedStandard	Indicates to the runtime that at most eight (8) bytes of the CLIENT_USER_ID and at most eight (8) bytes of the CLIENT_PASSWORD data will be set into the CFB header. The associated request buffer will not contain a CFB Security Offset area, and will therefore, not contain a security token. Additionally, by not making use of the CFB Security Offset area, the Client User ID and Client Password values are not eligible for being encrypted.
SecurityUsedEnhanced	Indicates to the runtime that the CLIENT_USER_ID, CLIENT_PASSWORD, and the optional Security Token should be added to the CFB by way of the CFB Security Offset. Additionally, at most (8) bytes of the Client User ID value will be set into the CFB header.
SecurityError	Indicates to the runtime that an error was encountered by the user exit and that the processing of the associated request has failed. The error indication and message string returned using the failureMsg argument would be returned to the GUI runtime. The GUI runtime will popup an error message display indicating the failed request.

Default Behavior

The WRSECTOKEN user exit, as delivered with CA Gen, will return SecurityNotUsed. In addition, although not necessary, the user exit will set the long integer pointed to by the tokenLen pointer to zero, and set the Boolean field pointed to by the bClntMgrSecurity pointer to False.

Building on Windows

The Client Security Token User Exit is built as part of the dynamic link library WRExx0N.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates the platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.

2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

3. Run MKEXITSN.BAT.

Related User Exits

The following are related user exits:

- TIRSECV
- WRSECENCRYPT
- WRSECDECRYPT

WRSRVRERROR—Server Flow Error Exit (Windows)

```
int WRSRVRERROR (int failureType,
    char *failureCommand,
    ErrorList errorList,
    ErrorToken errorToken)
```

Source Code

WREXITN.C

Purpose

This user exit is invoked when an error is detected during the processing of a synchronous client to server flow. The parameter failureType describes the origin of this error.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
failureType	Input	An integer value describing the source of the failure. It's value can be one of the following: "CFBUILD" is an error in the construction or parsing of a client/server flow message or response. "XFAL" identifies an error during the server procedures action block execution. "XERR" identifies a communications error occurring somewhere between construction of a message or response, and the deciphering of that message by the partner in this flow.
*failureCommand	Output	A character array, containing a maximum of 9 characters including the terminating NULL. This array can be populated with a command used to reinvoke the failing procedure step. This command is only used when returning from the exit with a FailAction of "serverFailedRestart." It will be ignored for any other FailAction.
errorList	Input/Output	An array of characters representing message strings constructed by and normally displayed using the ErrorReport dialog to describe the failure. Each message string is null terminated. Newline characters for formatting are also present as required. The complete list is terminated by more-than-one contiguous null character. On a "serverFailedDisplay" return, errorList, as modified in this exit, will be displayed in the ErrorReport dialog; errorList has a maximum length of 2048 bytes.
errorToken	Input	An array of characters, errorToken is only used with XFAL messages; errorToken can contain a token constructed by the Error Logging exit (TIRELOG) linked with the server load module describing the server action block failure; errorToken has a maximum length of 4097 bytes.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
serverFailedDisplay	This return value causes the standard error report dialog to be displayed, with return to the previous window.
serverFailedRestart	This return value suppresses the standard error report and reinvokes the client procedure step that originated the dialog flow. In the case of a failing "procedure step usage," the parent procedure step is returned to at the statement immediately following the Use. In both cases, if failureCommand is set it is used as the system command when reinvoking or returning to the client procedure step. For flows designed to return the server's exit state to the client, the exit state set when reinvoking or returning to the client procedure step will be the last value set by the client.
serverFailedTerminate	This return value will suppress the standard error report dialog, will not attempt to return to the client procedure step, and will redisplay the client window.

Default Behavior

The default return value of serverFailedDisplay causes the standard error report dialog to be displayed, with return to the previous client window.

Building on Windows

The Server Flow Error User Exit is built as part of the dynamic link library WRExx0N.DLL,

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates the platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.

2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VS for Visual Studio).

   Note: xx refers to the current release of CA Gen. abc refers to the supported version of Visual Studio. For the current release number and the supported version of Visual Studio, see the *Release Notes*.

3. Run MKEXITSN.BAT.

Related User Exits

The following are related user exits:

- TIRELOG
- WRASYNCSRVRERROR

WRSTRNCM—String Comparison Exit (Windows)

```
int WRSTRNCM (
    char *arg1,
    char *arg2,
    size_t len)
```

Source Code

WREXITN.C

Purpose

WRSTRNCM is called to compare two strings when the standard C language strncmp is not sufficient to deal with.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*arg1	Input	A pointer to a string to be compared.
*arg2	Input	A pointer to a second string to be compared.
Len	Input	Length of comparison.

Return Code

Integer containing comparison result. Returns 0 of strings match, returns a positive value if arg1 is greater than arg1, or returns a negative value if arg2 is greater than arg1.

Default Behavior

The default is to call the Windows function "strncmp".

Building on Windows

The String Comparison User Exit is built as part of the dynamic link library WRExx0N.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates the platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio 2010).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITSN.BAT.

Related User Exits

None

WRSYSID—System ID Exit (Windows)

```
int WRSYSID (
    char *rp1,
    char *rp2,
    char *systemid)
```

Source Code

WREXITN.C

Purpose

WRSYSID supplies the value for the LOCAL_SYSTEM_ID attribute. LOCAL_SYSTEM_ID can be placed on a window during window design. The value can be up to 8 characters in length.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*systemid	Output	A pointer to the system id.

Return Code

None

Default Behavior

The default value supplied is "WIN32."

Building on Windows

The System ID User Exit is built as part of the dynamic link library WRExx0N.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates the platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITSN.BAT.

Related User Exits

WRUSRID, TIRSYSID

WRTERMID—Terminal ID Exit (Windows)

```
int WRTERMID (
    char *rp1,
    char *rp2,
    char *termid)
```

Source Code

WREXITN.C

Purpose

WRTERMID supplies the value for the TERMINAL_ID attribute. TERMINAL_ID can be placed on a window during window design, and referenced by statements in a PrAD. The value can be up to 8 characters in length.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*termid	Output	A pointer to the terminal id.

Return Code

None

Default Behavior

The default value is "DOMAIN".

Building on Windows

The Terminal ID User Exit is built as part of the dynamic link library WRExx0N.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITSN.BAT.

Related User Exits

None

WRUPPR—Uppercase Translation Exit (Windows)

```
int WRUPPR (
    char *rp1,
    char * rp2,
    char *data)
```

Source Code

WREXITN.C

Purpose

WRUPPR is called when the runtime needs to use uppercase in a string.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*data	Input/Output	A pointer to the text to be uppercased.

Return Code

None

Default Behavior

The default is to call the Windows function "CharUpper," which handles uppercase characters.

Building on Windows

The Uppercase Translation User Exit is built as part of the dynamic link library WRExx0N.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITSN.BAT.

Related User Exits

TIRUPPR

WRURTL—Ultimate Retry Limit Exit (Windows)

```
long WRURTL ()
```

Source Code

WREXITN.C

Purpose

WRURTL specifies the maximum value for the TRANSACTION RETRY LIMIT system attribute. A SET TRANSACTION RETRY LIMIT statement in an action diagram and Default Retry Limit Exit cannot exceed this value.

Note: For more information about using the TRANSACTION RETRY LIMIT system attribute, see Default Retry Limit Exit (WRDRTL) (see page 74).

After the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, the application fails with a runtime error .

Arguments

None

Return Code

Long containing the retry limit.

Default Behavior

If the Ultimate Retry Limit User Exit is not used, the maximum value of TRANSACTION RETRY LIMIT will be 99 for all target environments. The Ultimate Retry Limit User Exit can be modified to return a value of zero to suppress all retry attempts.

Building on Windows

The Ultimate Retry Limit User Exit is built as part of the dynamic link library WRExx0N.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITSN.BAT.

Related User Exits

WRDRTL, TIRURTL

WRUSRID—User ID Exit (Windows)

```
int WRUSRID (
    char *rp1,
    char *rp2,
    char *userid)
```

Source Code

WREXITN.C

Purpose

WRUSRID supplies the value for the USER_ID attribute. USER_ID can be placed on a window during window design, and referenced by statements in a PrAD. The value can be up to 8 characters in length.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*userid	Output	A pointer to the user id.

Return Code

None

Default Behavior

The default value is "USERID".

Building on Windows

The User ID User Exit is built as part of the dynamic link library WRExx0N.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITSN.BAT (%GENxx%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITSN.BAT.

Related User Exits

WRSYSID, TIRUSRID

Windows Client Middleware User Exits

Client Manager - Windows User Exits

All supplied Client Manager user exits are written using the C programming language. The following table briefly describes the Client Manager Exits:

Client Manager: Language: C		
User Exit Name	**Source Code**	**Description**
CI_CM_DPC_FLOW_COMPLETE_COMM_ERROR	cicmclx.c	Handle communications error while processing a synchronous cooperative flow request.
CI_CM_ID	cicmclx.c	Client Manager unique ID (supports the use of Multi-Instance Client Manager). Used by the Client Manager application.
CIDE_INIT	cidexit.c	Conversation Instance Data—Initialize. Used to disable or enable subsequent CIDE_PROC calls.
CIDE_PROC	cidexit.c	Conversation Instance Data—Process. Used to modify certain fields of the Conversation Instance data (for example UserId and Password), prior to the conversation supporting a cooperative flow being created.
DECRYPT	decrexit.c	Decrypt CFB from GUI client if the data in Enhanced Security is to be used and the target server environment has a derived Security_Level of Remote and the CFB encryption flag indicates the CFB has been encrypted.
IEFDP_CLEANUPDIR	iefdir.c	Client Manager Directory Services—Cleanup Allows for deallocation of resources, which can have been allocated to support directory services.

Client Manager: Language: C		
User Exit Name	**Source Code**	**Description**
IEFDP_INITDIR	iefdir.c	Client Manager Directory Services—Initialize (disable or enable subsequent Directory services calls)
IEFDP_SEARCHDIR	iefdir.c	Client Manager Directory Services—Search Implementation of the transaction server search algorithm.
RSCUSERENTRY	iorscclx.cxx	This exit supports multiple APIs that allow inspection and modification of user data and application data before it is sent to a target server on HP NonStop. User and application data received from the target server can also be inspected or modified before forwarding to the client application.

Details for the preceding user exits follow in a separate section for each.

CI_CM_DPC_FLOW_COMPLETE_COMM_ERROR – Client Manager Communications Error Exit (Windows)

```
intCI_CM_DPC_Flow_Complete_Comm_Error (
    int numberOfAttempts,
    unsigned long resonCode )
```

Source Code

CICMCLX.C

Purpose

Control is passed to this exit if the Client Manager encounters a communications error while processing a synchronous cooperative flow request.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
numberOfAttempts	Input	This integer represents the number of attempts that will be made for a particular reason code.
reasonCode	Input	This integer represents the error that was encountered.

Return Code

Return Code	Description
RETRY_OK	Retry the processing of the cooperative flow.
RETRY_NOT_OK	Terminate processing of the cooperative flow.

Default Behavior

Returns a value of RETRY_NOT_OK.

Building the Exit

The CI_CM_DPC_Flow_Complete_Comm_Error exit is built as a part of the dynamic link library CMICXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CCMEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CCMEXIT.NT CLEAN.

5. Run NMAKE /F CCMEXIT.NT.
6. Copy the CMICXxxN.DLL user exit into the Client Manager's install directory.

Related User Exits

None

CI_CM_ID—Client Manager ID Exit (Windows)

```
char * ci_cm_id( void );
```

Source Code

CICMCLX.C

Purpose

To support multiple instances of the Client Manager, you must modify the Client Manager ID user exit. This user exit lets a unique Windows IPC API mailslot name be created for each instance of the Client Manager that executes in a multi-user environment. Those clients expecting to connect to this same Client Manager instance must also use this unique mailslot name used by a given Client Manager instance. The matching of mailslot naming is accomplished because both the Client Manager and the Client Manager cooperative flow runtime code will utilize the same Multi-Instance user exit dll.

To enable the multi-instance capability of the Client Manager, the customer must modify this exit routine. Any mechanism for determining unique strings logged on user basis can be used.

Note: For more information about multi-instance Client Managers, see the *Distributed Processing—Client Manager User Guide*.

After modification, both the CA Gen client, using the client manager cooperative flow runtime dll, and the Client Manager, use this user exit.

Arguments

None

Return Code

Returns a unique string identifier used to create/identify a Client Manager's mailslot on a Windows workstation.

Default Behavior

Returns a value of TIRCLNTS.

Building the Exit

The ci_cm_id exit is built as a part of the dynamic link library CMICXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CCMEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CCMEXIT.NT CLEAN.
5. Run NMAKE /F CCMEXIT.NT.
6. Copy the CMICXxxN.DLL user exit into the Client Manager's install directory.

Steps to Validate Successful Incorporation of the CMICXnnN.DLL

Prior to incorporating the modified exit, follow these steps:

1. Start the Client Manager.
2. Set Client Manager Logging Level to Trace, using the Client Manager's Setup dialog (that is, from the Client Manager's main menu select File, Setup). Save the configuration change and exit the Client Manager.
3. Restart the Client Manager.
4. Browse the Client Manager Log. Look for the log file record containing the name of the client queue. The default Client Manager mailslot name will contain the following string: .\mailslot\TIRCLNTS.QUE
5. Stop the Client Manager.

6. After modifying and rebuilding the modified user exit dll, restart the Client Manager.
7. Browse the Client Manager Log. The client queue displayed should be of the form: .\mailslot\xxxxxxxx.QUE, where *xxxxxxxx* is the string value returned from the modified user exit. This string should be unique to each instance of the Client Manager that is expected to run.

After modification, both the Client Manager instance and all of the CA Gen clients expecting to connect to this instance of the Client Manager must use this user exit.

Related User Exits

None

CIDE_INIT—Conversation Instance Data Initialize Exit (Windows)

```
int CIDE_INIT (void);
```

Source Code

CIDEXIT.C

Purpose

For those transports that use UserID and Password as part of their protocol (currently LU6.2) a set of user exit functions is provided to facilitate any required adjustments of the UserID and Password prior to being sent to the transport layer. The CPI/C API performs the ASCII to EBCDIC translation of the UserID and Password as part of its conversation protocol. There are two entry points associated with this user exit:

This, the first entry point, is invoked only once, during initialization of the Client Manager transport support code. Future calls to the CIDE_PROC user exit will be enabled or disabled depending upon the return value from this exit.

Arguments

None

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
CIDExitDisabled	The default exit implementation disables all future calls into this exit. The CIDE_PROC user exit will never be invoked.

Return Code	Description
CIDExitEnabled	Return this value to enable subsequent processing of the Conversation Instance Data Exit entry point, .CIDE_PROC.

Default Behavior

This exit by default disables all future calls into the CIDE_PROC user exit.

Building the Exit

The Conversation Instance Data User exit is built as a part of the dynamic link library CIDExxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MAKECID.BAT (%GEN*xx*%Gen\VSabc\samples\ClientManager for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Execute the bat file: MAKECID.BAT.
4. Copy the CIDExxN.DLL user exit into the Client Manager installation directory.

Related User Exits

CIDE_PROC

CIDE_PROC—Conversation Instance Data Process Exit (Windows)

```
void CIDE_PROC (unsigned short LocalCodePage,
    unsigned short RemoteCodePage,
    unsigned short DataCodePage,
    unsigned char * TranCode,
    unsigned char * UserId,
    unsigned char * Password);
```

Source Code

CIDEXIT.C

Purpose

For those transports that use UserID and Password as part of their protocol (currently LU6.2) a set of user exit functions is provided to facilitate any required adjustments of the UserID and Password prior to being sent to the transport layer. The CPI/C API performs the ASCII to EBCDIC translation of the UserID and Password as part of its conversation protocol. There are two functions associates with this user exit:

This second entry point lets you modify the UserID and Password fields before they are passed to the transport protocol layer. This can be used for codepage translation of the UserID and Password, if required by the DPS, prior to the cooperative flow.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
LocalCodePage	Input	CodePage value associated with the client workstation
RemoteCodePage	Input	CodePage value associated with the target server (value as defined in the .srv file)
DataCodePage	Input	CodePage value associated with the following data items host name
*TranCode	Input	Pointer to a null terminated string containing the conversation instance trancode value
*UserId	Input/Output	Pointer to a null terminated string containing the conversation instance user ID value. Modifications are propagated to the cooperative flow request.
*Password	Input/Output	Pointer to a null terminated string containing the conversation instance password value. Modifications are propagated to the cooperative flow request.

Return Code

None

Default Behavior

CIDE_PROC is not invoked.

Building the Exit

The Conversation Instance Data User exit is built as a part of the dynamic link library CIDExxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MAKECID.BAT (%GEN*xx*%Gen\VSabc\samples\ClientManager for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Execute the bat file: MAKECID.BAT.
4. Copy the CIDExxN.DLL user exit into the Client Manager installation directory.

Related User Exits

CIDE_INIT

DECRYPT—Cooperative Flow Decryption Exit (Windows)

```
int DECRYPT (long maxViewLen,
    long *encryptViewLen,
    unsigned char *encryptView,
    char *failureMsg);
```

Source Code

DECREXIT.C

Purpose

The CFB data transmitted from a DPC application can optionally be encrypted. A flag byte in the CFB header notifies the receiver of the CFB that the CFB has been encrypted. It is the receiver's responsibility to decrypt the CFB prior to using it.

The runtime's WRSECTOKEN user exit controls if and where within the CFB the client application's security data is placed. If the WRSECTOKEN user exit returns SecurityUsedEnhanced, the client's security data will be placed in the security offset section of the CFB. This portion of the CFB can optionally be encrypted by the client runtime's WRSECENCRPT user exit.

If the derived security level of the selected target server is Remote, the Client Manager can need to decrypt the CFB to access the security data placed into the security offset section of the CFB.

The Client Manager will invoke its DECRYPT user exit if each of the following conditions is met:

1. The derived security level of the selected target server is Remote. Remote indicates that the Client Manager will attempt to associate security data for flows that target this server.
2. The CFB Being processed is an Enhanced CFB indicating that the CFB contains a security offset area.
3. The CFB's bClntMgrSecurity flag is set. This flag is set by the client runtime if the client's WRSECTOKEN user exit sets its bClntMgrSecurity argument to TRUE. This CFB flag informs the Client Manager that it should use the security data contained within the security offset area of the CFB rather than the security data it maintains as part of its configuration.
4. The CFB has been encrypted by the client runtime. The DPC encrypts the CFB using the WRSECENCRYPT user exit. If the client encrypts the CFB, then the Client Manager must decrypt the CFB to extract the security data from the security offset area.

Client Manager's DECRYPT user exit will not be invoked if:

- The target server's derived security level is None.
- The CFB being processed by the Client Manager is a Standard CFB. In this case, the CFB does not contain a security offset.
- The CFB being processed is not encrypted.

Note: For more information about security configuration, see the *Distributed Processing—Client Manager User Guide*.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
MaxViewLen	Input	A long field that contains the maximum available buffer space (in bytes) that the decrypted data can occupy.
*EncryptViewLen	Input/Output	Input: the current buffer space (in bytes) of the encrypted data. Output: Len should be updated to contain the length of the decrypted data. The length of the decrypted result cannot exceed maxViewLen.
*EncryptView	Input/Output	A pointer to a conversion work area. Input: The work area contains the encrypted data. Output: This exit must update the work area to contain the decrypted version of the encrypted data.
*failureMsg	Output	A pointer to an 80-character array that can receive a user provided null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into error text returned to the DP client.

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
DecryptionNotUsed	Indicates to the runtime that the user exit did not perform any decryption of the data provided. This is the default return value.
DecryptionUsed	Indicates to the runtime that the user exit successfully performed the decryption of the provided data.
DecryptionFailure	Indicates to the runtime that an error was encountered by the user exit and that the decryption processing has failed. The error indication and message string returned using the failureMsg argument will be returned to the DP client associated with the failed request.

Default Behavior

The DECRYPT user exit, as delivered, does not perform decryption and will return DecryptionNotUsed.

Building the Exit

The Decrypt exit is built as a part of the dynamic link library DECRExxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MAKEDECR.BAT (%GEN*xx*%Gen\VSabc\samples\ClientManager for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Execute the bat file: MAKEDECR.BAT.
4. Copy the DECRExxN.DLL user exit into the Client Manager installation directory.

Related User Exits

The following are related user exits:

- WRSECTOKEN
- WRSECENCRYPT

IEFDP_CLEANUPDIR—Directory Services Cleanup Exit (Windows)

```
void IEFDP_CleanUpDir (void);
```

Source Code

IEFDIR.C

Purpose

The Directory Services User Exit is the Client Manager's implementation of Transaction routing. Transaction routing is a conceptual process that lets cooperative flow data be routed from a Distributed Process Client (DPC) to a programmatically determined Distributed Process Server (DPS). The *Distributed Processing—Client Manager User Guide* discusses transaction routing and directory services in detail.

IEFDP_CleanUpDir is called when Client Manager is terminating. It takes no arguments and does not return a value. In the sample code, IEFDP_CleanUpDir is used to free any resources allocated on behalf of the Client Manager. IEFDP_CleanUpDir is the last directory service function called by the Client Manager.

Arguments

None

Return Code

None

Default Behavior

The Client Manager Directory Services transaction routing is disabled.

Building the Exit

The Directory Services exit is built as a part of the dynamic link library IEFDIRN.DLL. A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile IEFDIRN.MAK (%GEN*xx*%Gen\VSabc\samples\ClientManager for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F IEFDIRN.MAK CLEAN.
5. Run NMAKE /F IEFDIRN.MAK.
6. Copy the IEFDIRN.DLL user exit into the Client Manager installation directory.

Note: Because the Directory Services user exit dll name is configurable using the Client Manager setup dialog, the dll file does not have to be named IEFDIRN.DLL. If the dll is renamed, the IEFDIRN.MAK file must be modified accordingly.

Related User Exits

The following are related user exits:

- IEFDP_InitDir
- IEFDP_SearchDir

IEFDP_INITDIR—Directory Services Initialize Exit (Windows)

```
unsigned long IEFDP_InitDir (void* iniFile);
```

Source Code

IEFDIR.C

Purpose

The Directory Services User Exit is the Client Manager's implementation of Transaction routing. Transaction routing is a conceptual process that lets cooperative flow data be routed from a Distributed Process Client (DPC) to a programmatically determined Distributed Process Server (DPS). The *Distributed Processing—Client Manager User Guide* discusses transaction routing and directory services in detail.

This initialization interface is called once and does the Client Manager call the first directory service function. In the supplied sample implementation, this routine reads the transaction code to server mapping information from the iefdir.trn file.

Note: For more information about updating the IEFDIR.TRN file, see the section Directory Services DLL Functions in the *Distributed Processing—Client Manager User Guide*.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
* iniFile	Input	A void pointer, pointing to a NULL terminated string containing the name of the Client Manager initialization file. (By default IEFCMN.INI).

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
Zero (0)	Indicates that the initialization was successful.

Return Code	Description
Non Zero	If a non-zero value is returned to the Client Manager, the Client Manager will assume that the directory services capability is not available.

Default Behavior

The Client Manager Directory Services transaction routing is disabled.

Building the Exit

The Directory Services exit is built as a part of the dynamic link library IEFDIRN.DLL. A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile IEFDIRN.MAK (%GEN*xx*%Gen\VSabc\samples\ClientManager for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F IEFDIRN.MAK CLEAN.
5. Run NMAKE /F IEFDIRN.MAK.
6. Copy the IEFDIRN.DLL user exit into the Client Manager installation directory.

Note: Because the Directory Services user exit dll name is configurable using the Client Manager setup dialog, it need not be called IEFDIRN.DLL. If the dll is to be renamed, the IEFDIRN.MAK file will need to be modified accordingly.

Related User Exits

The following are related user exits:

- IEFDP_SearchDir
- IEFDP_CleanUpDir

IEFDP_SEARCHDIR—Directory Services Search Exit (Windows)

```
void* IEFDP_SearchDir (void *parms);
```

Source Code

IEFDIR.C

Purpose

The Directory Services User Exit is the Client Manager's implementation of Transaction routing. Transaction routing is a conceptual process that lets cooperative flow data be routed from a Distributed Process Client (DPC) to a programmatically determined Distributed Process Server (DPS).

Note: For more information about transaction routing and directory services, see the *Distributed Processing—Client Manager User Guide.*

This exit is used to return a target system name to the Client Manager. This exit takes a single argument which points to two (2) NULL terminated strings. The first string is the transaction code and the second string is NextLocation data provided by the application. IEFDP_SearchDir should return a NULL value to the Client Manager if an appropriate target system cannot be found. If NextLocation data is provided, the sample program assumes that it contains a target system name.

Each time IEFDP_SearchDir is called, the sample code determines if the iefdir.trn input file has changed. If it has, the binary search tree is refreshed.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*parms	Input	A void pointer, pointing to 2 NULL terminated strings. The first string contains a transaction code, the string immediately following the first string's terminating NULL is the NEXTLOCATION provided by the application.

Return Codes

The return code is briefly described next.

void *	A pointer to a string containing one of the following: ■ The NEXTLOCATION string if it was provided to this exit by the application. ■ A target system name as obtained from the implementation of this interface. In this illustration, the target system name is obtained from the IEFDIR.TRN file. ■ A NULL of an appropriate target system name cannot be found.

Note: The sample implementation of this exit will attempt to remove trailing spaces from the string pointed to by this void *. Thus, this pointer should not refer to read-only memory. You can modify this exit's sample function clipString(), located in iefdir.c if required to accommodate the void * pointing to read-only memory.

Default Behavior

The Client Manager Directory Services transaction routing is disabled.

Building the Exit

The Directory Services exit is built as a part of the dynamic link library IEFDIRN.DLL. A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile IEFDIRN.MAK (%GEN*xx*%Gen\VSabc\samples\ClientManager for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F IEFDIRN.MAK CLEAN.
5. Run NMAKE /F IEFDIRN.MAK.
6. Copy the IEFDIRN.DLL user exit into the Client Manager installation directory.

Note: Because the Directory Services user exit dll name is configurable using the Client Manager setup dialog, it need not be called IEFDIRN.DLL. If the dll is to be renamed, the IEFDIRN.MAK file will need to be modified accordingly.

Related User Exits

The following are related user exits:

- IEFDP_InitDir
- IEFDP_CleanUpDir

RSCUSERENTRY—Entry Point for Accessing APIs for User/Application Data Targeting HP NonStop Servers (Windows)

```
int RSCUserEntry(pTRANSHANDLE const transHandle, int sending );
```

Source Code

IORCSCLX.CXX

Purpose

The RSCUserEntry user exit supports multiple APIs that let you inspect and modify user data and application data before it is sent to the target server. User and application data received from the target server can also be inspected or modified before forwarding to the client application.

Note: The size of the CA Gen data buffer cannot be modified. Data integrity must be maintained by the user exit.

Each user exit API may be called just before sending data to the target server and again just after receiving data from the target server. This enables use of the APIs for such purposes as encrypting/decrypting transaction data, adding custom non Gen data to the buffer sent to the server side, and performing customized auditing.

More information:

API Functions (see page 136)

APIs

The following table describes of each of the APIs supported by the Remote Server Call interface, RSCUserEntry:

API Name	Description
GetMessageSize	Returns the length of the message area for the current request or response message.
GetUserData	Returns the user data associated with the current request or response message.

API Name	Description
SetUserData	Copies the passed in buffer into the user data area associated with the current request or response message.
GetIEFData	Returns the CA Gen data (and its length) contained within the current receive or response message.
SetIEFData	Modifies the CA Gen data contained within the buffer used for the current receive or response message.

Return Code

This user exit returns True if the API function succeeds, or returns False if the API function fails.

Default Behavior

The RSCUserEntry is invoked only when the user exit .dll file is configured using the Client Manager RSC/MP Configuration Details dialog.

Building the Exit

The RSC/MP User exit is built as a part of the dynamic link library RSCUXxxN.dll.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

The Microsoft's Visual C++ compiler should be installed on the system where you build the .dll.

Follow these steps:

1. Launch a Command Prompt window.
2. Change to the directory that contains the makefile IORSCUX.NT (%GEN*xx*%Gen\VSabc\samples\ClientManager for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F IORCSUX.NT.
5. Copy the RSCUXxxN.DLL user exit into the Client Bridge installation directory.

Related User Exits

USEREXIT—RSC/MP Server Side User Exit

API Functions

This section provides information about how to use the API Functions of the RSCUserEntry user exit.

Function Format

```
int GetMessageSize( pTRANSHANDLE transHandle, short * msgSize )
```

Purpose

The GetMessageSize API function returns the size of the current message buffer. This includes the NonStop RSC/MP header, Gen data, and user data.

If the second parameter passed into the main entry point, RSCUserEntry(), is 1 this signifies the exit is being called just prior sending the request message to the target server. The message size returned will be that of the message about to be sent to the server.

If the second parameter passed into the main entry point, RSCUserEntry(), is 0 this signifies the exit is being called just after the response message has been received from the target server. The message size returned will be that of the message just received from the server.

Arguments

The following table describes the arguments for the GetMessageSize API Function:

Name	I/O	Description
transHandle	Input	The transaction handle for this session. This value is passed into the RSCUserEntry entry point by the Gen runtime. This value must not be changed.
msgSize	Output	A pointer to a short where the total size of the current message is stored.

Return Code

The following table describes each return code for the GetMessageSize API function:

Return Code	Description
0	The API function succeeded.
Nonzero	An invalid transHandle was detected.

Default Behavior

When enabled, the default RSCUserEntry user exit calls this API after receiving a response from the server, prior to forwarding it to the client.

Function Format

```
int GetUserData( pTRANSHANDLE transHandle, unsigned char * data, short * length )
```

Purpose

Returns the user data area associated with the current message. If the second parameter passed into the main entry point, RSCUserEntry(), is 1 then this signifies the exit is being called just before sending the request message to the target server. The user data returned is the data previously set by the client, if any.

If the second parameter passed into the main entry point, RSCUserEntry(), is 0 then this signifies the exit is being called just after the response message has been received from the target server. The user data returned will be that previously set by the server, if any.

Arguments

The following table describes the arguments for the GetMessageSize API Function:

Name	I/O	Description
transHandle	Input	The transaction handle for this session. This value is passed into the RSCUserEntry entry point by the Gen runtime. This value must not be changed.
* data	Input/ Output	A pointer to a buffer area where the runtime is to copy the user data. **Note:** It is the caller's responsibility to ensure adequate memory has been allocated to contain the user data contained in the current message.
* length	Output	A pointer to a short that will contain the actual size of the data copied.

Return Code

The following table describes each return code for the GetUserData API function.

Return Code	Description
0	The API function succeeded.
Nonzero	An invalid transHandle was detected.

Default Behavior

If enabled, the default RSCUserEntry user exit calls this API after having received a response from the server, prior to forwarding it to the client.

Function Format

```
int SetUserData( pTRANSHANDLE transHandle, unsigned char * data, short * length )
```

Purpose

Copies the passed in buffer into the user data area associated with the current message. C–104 Distributed Processing-Overview Guide

If the second parameter passed into the main entry point, RSCUserEntry(), is 1 then this signifies the exit is being called just prior sending the request message to the target server. The buffer data will be copied into the current message's user data area prior to the message being sent to the target server.

If the second parameter passed into the main entry point, RSCUserEntry(), is 0 then this signifies the exit is being called just after the response message has been received from the target server. Calling this API at this time will effectively have no influence on the current message.

Arguments

The following table describes the arguments for the SetUserData API Function:

Name	I/O	Description
transHandle	Input	The transaction handle for this session. This value is passed into the RSCUserEntry entry point by the Gen runtime. This value must not be changed.
* data	Input/ Output	A pointer to a buffer area that is copied into the current message's user data area.
length	Output	The length of the data buffer pointed to by the data parameter.

Return Code

The following table describes each return code for the SetUserData API function:

Return Code	Description
0	The API function succeeded.
Nonzero	An invalid transHandle was detected.

Default Behavior

If enabled, the default RSCUserEntry user exit calls this API prior to the request being sent to the server.

Function Format

```
int GetIEFData( pTRANSHANDLE transHandle, short sending, unsigned char * data, short * length )
```

Purpose

Returns a copy of the client data associated with the current message.

If the second parameter passed into the main entry point, RSCUserEntry(), is 1 the this signifies the exit is being called just prior sending the request message to the target server. The client data returned will be that which will be forwarded on to the server.

If the second parameter passed into the main entry point, RSCUserEntry(), is 0 then this signifies the exit is being called just after the response message has been received from the target server. The data returned will be that which will be forwarded on to the client.

Arguments

The following table describes the arguments for the SetUserData API Function:

Name	I/O	Description
transHandle	Input	The transaction handle for this session. This value is passed into the RSCUserEntry entry point by the Gen runtime. This value must not be changed.
Sending	Output	This value must be set the same as the second parameter passed into RSCUserEntry() by the runtime.
* data	Input/ Output	A pointer to a buffer area where the runtime is to copy the client data. **Note:** It is the caller's responsibility to ensure that adequate memory has been allocated to contain the client data copied from the current message.
* length	Output	A pointer to a short that will contain the actual size of the data copied.

Return Code

Return Code

The following table describes each return code for the SetUserData API function:

Return Code	Description
0	The API function succeeded.
Nonzero	An invalid transHandle was detected.

Default Behavior

If enabled, the default RSCUserEntry user exit calls this API just before sending the request to the server and calls it again just after the response has been returned from the server.

Function Format

```
int SetIEFData( pTRANSHANDLE transHandle, short sending, unsigned char * data )
```

Purpose

Copies the passed in buffer into the client data area associated with the current message.

If the second parameter passed into the main entry point, RSCUserEntry(), is 1, then this signifies that the user exit is being called just before sending the request message to the target server. The buffer data will be copied into the current message's client data area prior to the message being forwarded to the target server.

If the second parameter passed into the main entry point, RSCUserEntry(), is 0, then this signifies that the user exit is being called just after the response message has been received from the target server. The buffer data will be copied into the current message's client data area prior to the message being forwarded to the client.

Arguments

Name	I/O	Description
transHandle	Input	The transaction handle for this session. This value is passed into the RSCUserEntry entry point by the Gen runtime. This value must not be changed.
Sending	Output	This value must be set the same as the second parameter passed into RSCUserEntry() by the runtime.

Name	I/O	Description
* data	Input/ Output	A pointer to a buffer area that is copied into the current message's client data area. The number of bytes copied will correspond to the client data size as know by the runtime. It is the caller's responsibility to maintain the integrity of the client data structure and size.

Return Code

The following table gives a brief description of each of the return code values:

Return Code	Description
0	Indicates that the API completed successfully.
Nonzero	Indicates that the API detected an invalid transHandle.

Default Behavior

If enabled, the default user exit calls this API just before sending the request to the server and again just after the response has been returned from the server.

Communications Bridge - Windows User Exits

All supplied Communications Bridge user exits are written using the C programming language. The following table briefly describes each of the exits:

Comm. Bridge: Language: C		
User Exit Name	**Source Code**	**Description**
CIDE_INIT	cidexit.c	Conversation Instance Data exit - Initialize This exit disables or enables subsequent CIDE_PROC calls.
CIDE_PROC	cidexit.c	Conversation Instance Data exit—Process This exit lets modification of certain fields of the Conversation Instance data (for example UserId and Password), prior to the conversation supporting a cooperative flow being created.

Comm. Bridge: Language: C		
User Exit Name	**Source Code**	**Description**
DECRYPT	decrexit.c	This exit will decrypt the CFB from a client if the data in the CFB's Enhanced Security offset area is to be used and the target server environment has a derived Security_Level of Remote and if the CFB has been encrypted.
ECI_CLIENT_EXIT	ioeciclx.c	This exit is called to permit overriding the name of the target CICS System (as it is known to the CICS Universal Client), the specified ECI timeout value, and the specified CICS Mirror Transaction associated with the request.
GETTCPHOSTNAME	inetipux.c	Allows disabling the host name lookup that maps a connected client's IP address to a host name string.
RSCUSERENTRY	iorscclx.cxx	This exit supports multiple APIs that allow inspection and modification of user data and application data before it is sent to a target server on HP NonStop. User and application data received from the target server can also be inspected or modified before forwarding to the client application.

Details of the preceding user exits follow. Each one is described in a separate section.

CIDE_INIT—Conversation Instance Data User Exit (Windows)

```
int CIDE_INIT (void);
```

Source Code

CIDEXIT.C

Purpose

For those transports that use UserID and Password as part of their protocol (currently LU6.2) a set of user exit functions is provided to facilitate any required adjustments of the UserID and Password prior to being sent to the transport layer. The CPI/C API performs the ASCII to EBCDIC translation of the UserID and Password as part of its conversation protocol. There are two entry points associated with this user exit:

This, the first, entry point is invoked only once, during initialization of the Client Manager transport support code. Future calls to the CIDE_PROC user exit will be enabled or disabled depending upon the return value from this exit.

Arguments

None

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
CIDExitDisabled	The default exit implementation disables all future calls into this exit. The CIDE_PROC user exit will never be invoked.
CIDExitEnabled	Return this value to enable subsequent processing of the Conversation Instance Data Exit entry point, .CIDE_PROC.

Default Behavior

This exit by default disables all future calls into the CIDE_PROC user exit.

Building the Exit

The Conversation Instance Data User exit is built as a part of the dynamic link library CIDExxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MAKECID.BAT (%GEN*xx*%Gen\VSabc\samples\CommBridge for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Execute the bat file: MAKECID.BAT
4. Copy the CIDExxN.DLL user exit into the Comm. Bridge installation directory

Related User Exits

None

CIDE_PROC—Conversation Instance Data Process Exit (Windows)

```
void CIDE_PROC (unsigned short  LocalCodePage,
    unsigned short  RemoteCodePage,
    unsigned short  DataCodePage,
    unsigned char * TranCode,
    unsigned char * UserId,
    unsigned char * Password);
```

Source Code

CIDEXIT.C

Purpose

The two user exit functions are provided to facilitate any required translation of the UserID and Password prior to being sent to the transport layer. This second entry point lets you modify the UserID and Password fields before they are passed to the transport protocol layer. This can be used for codepage translation of the UserID and Password, if required by the DPS, prior to the cooperative flow.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
LocalCodePage	Input	CodePage value associated with the client workstation
RemoteCodePage	Input	CodePage value associated with the target server (value as defined in the .srv file)
DataCodePage	Input	CodePage value associated with the following data items host name
*TranCode	Input	Pointer to a null terminated string containing the conversation instance trancode value
*UserId	Input/Output	Pointer to a null terminated string containing the conversation instance user ID value. Modifications made to the field pointed to by UserId are propagated to the CFB associated with the cooperative flow request.
*Password	Input/Output	Pointer to a null terminated string containing the conversation instance password value. Modifications made to the field pointed to by Password are propagated to the CFB associated with the cooperative flow request.

Return Code

None

Default Behavior

CIDE_INIT by default disables all calls into CIDE_PROC .

Building the Exit

The Conversation Instance Data Process User exit is built as a part of the dynamic link library CIDExxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MAKECID.BAT (%GEN*xx*%Gen\VSabc\samples\CommBridge for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Execute the bat file: MAKECID.BAT
4. Copy the CIDExxN.DLL user exit into the Comm. Bridge installation directory

Related User Exits

None

DECRYPT—Cooperative Flow Decryption Exit (Windows)

```
int DECRYPT (long maxViewLen,
    long *encryptViewLen,
    unsigned char *encryptView,
    char *failureMsg);
```

Source Code

DECREXIT.C

Purpose

The CFB data transmitted from a DPC application can optionally be encrypted. A flag byte in the CFB header notifies the receiver of the CFB that the CFB has been encrypted. It is the receiver's responsibility to decrypt the CFB prior to using it.

With respect to the Comm. Bridge, the data in the CFB will only need to be decrypted if the security data located in the security offset section is to be used when sending the cooperative flow request to a target server. The Comm. Bridge uses the CMUseSecure CFB flag to determine if the data in the security offset should be used.

The CMUseSecure flag is set by the client runtime if its invocation of the client side security exit returns a TRUE for the bClntMgrSecurity flag. Therefore, the following considerations must be met before the Comm. Bridge invokes its decryption user exit:

1. The CFB's CMUseSecure flag is set, indicating that the requesting client requires the Comm. Bridge use the security data it placed into the security offset section when handling the cooperative flow.
2. The CFB encryption flag byte indicates that it has been encrypted.

If the preceding conditions are met, the Comm. Bridge must invoke its decryption user exit.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
MaxViewLen	Input	A long field that contains the maximum available buffer space (in bytes) that the decrypted data can occupy.
*EncryptViewLen	Input/ Output	Input: the current buffer space (in bytes) of the encrypted data. Output: Len should be updated to contain the length of the decrypted data. The length of the decrypted result cannot exceed maxViewLen.
*EncryptView	Input/ Output	A character pointer to a buffer containing view data. On input, the buffer contains the encrypted data. On output, this exit should ensure the buffer contains the decrypted version of the encrypted data.
*failureMsg	Output	A pointer to an 80-character array that can receive a user provided null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into error text returned to the DP client.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
DecryptionNotUsed	Indicates to the runtime that the user exit did not perform any decryption of the data provided. This is the default return value.

Return Code	Description
DecryptionUsed	Indicates to the runtime that the user exit successfully performed the decryption of the provided data.
DecryptionFailure	Indicates to the runtime that an error was encountered by the user exit and that the decryption processing has failed. The error indication and message string returned using the failureMsg argument will be returned to the DP client associated with the failed request.

Default Behavior

The DECRYPT user exit, as delivered, does not perform decryption and will return DecryptionNotUsed.

Building the Exit

The Decrypt exit is built as a part of the dynamic link library DECRExxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MAKEDECR.BAT (%GEN*xx*%Gen\VSabc\samples\CommBridge for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Execute the bat file: MAKEDECR.BAT
4. Copy the DECRExxN.DLL user exit into the Comm. Bridge installation directory

Related User Exits

The following are related user exits:

- TIRELOG
- WRSRVRERROR

ECI_CLIENT_EXIT—ECI Communications Interface Exit (Windows)

```
short eci_client_exit( char  * pTranCode,
    short * pECITimeOut,
    char  * pECITpn,
    char  * pSysName );
```

Source Code

IOECICLX.C

Purpose

A Comm. Bridge can be configured to use ECI as its server side transport. The ECI runtime lets applications target one or more CICS regions. This user exit is called from the Comm. Bridge's ECI runtime DLL prior to it invoking the CICS ExternalCall() API. Invoking this exit lets the ECI runtime's default behavior be customized. The name of the target CICS region, the ECI timeout value and the TP Name to use when processing the request can all be customized within this exit. If the ECI TPN is specified by this exit, the name specified must be associated with the CICS Mirror application DFHMIRS.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
* pTranCode	Input	A pointer to a character array containing the transaction code name associated in the model with the target server procedure step. Some users can wish to set the ECI TPN to the value pointed to by this input argument.
* pECITimeOut	Input/Output	Input: pointer to a signed short containing the default timeout value. 0 (zero) seconds indicates no timeout. Output: the signed short field pointed to by this pointer is used as the eci_timeout value specified in the ECI_PARMS structure used by the CICS ExternalCall() API call.
* pECITpn	Input/Output	Input: A pointer to a NULL 4-byte character array. Output: A NULL character array results in CPMI being used as the ECI TPN on the CICS ExternalCall(). A non-NULL 4-byte character array contains the eci_tpn value that will be specified in the ECI_PARMS structure used by the CICS ExternalCall() API call .

Name	I/O	Description
* pSysName	Input/Output	Input: Pointer to a character array containing a default CICS System name. Output: The memory area pointed to by this address will contain the name of the CICS system to be used on the ECI API call. The character array is defined to receive a null terminated string whose maximum length is CICS_ECI_SYSTEM_MAX+1.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
0 (Zero)	The operation of the exit completed OK
!= 0 (Non zero)	An error occurred during the processing of the exit.

Default Behavior

The default user exit returns a zero. The default behavior does not attempt to modify any of the customizable fields.

Building the Exit

The eci_client_exit is built as a part of the dynamic link library ECIUXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile IOECIUX.NT (%GEN*xx*%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F IOECIUX.NT CLEAN.
5. Run NMAKE /F IOECIUX.NT.

Related User Exits

None

GETTCPHOSTNAME—Host Name Lookup Exit (Windows)

```
void GetTCPHostName (sockaddr_storage sock_addr,
    char * hostName,
    int maxHostNameLen);
```

Source Code

INETIPUX.C

Purpose

This user exit is used to disable the domain name server (DNS) host name lookup for clients that connect to a Comm. Bridge. Host name lookup is normally performed when adding a client's host name to the Client Name list on the Comm. Bridge main window. In general, it is usually easier to deal with a client's host name than it is its IP address. By default, the Comm. Bridge will attempt to display its client connections using their host names.

The host name lookup is accomplished by calling the system routine getnameinfo(). In certain network environments, the time required to obtain a host name from an IP address can be excessive, especially when the domain name server is behind a firewall. The time to resolve the host name from its IP address is overhead expended when a client connects to the Comm. Bridge. The client can see this overhead as its server application taking longer than expect to return its response.

This exit gives a mechanism whereby the getnameinfo() call can return either a text host name or an IP address. If the IP address of the client is returned then the IP address is displayed in the Client Name list.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
In	Input	A socket sockaddr_storage structure containing the IP address for which a textual hostname is to be provided.

Name	I/O	Description
*hostName	Output	A pointer to a character array updated by this exit with the hostname corresponding to the sockaddr_storage structure content.
maxHostNameLen	Input	An integer value containing the maximum length of the character array pointed to by the hostName argument.

Return Code

None

Default Behavior

A getnameinfo() function call is issued to lookup a given clients IP address and to obtain its text hostname equivalent.

Building the Exit

The GetTCPHostName exit is built as a part of the dynamic link library TCPUXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. *N* indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile INETIPUX.NT (%GEN*xx*%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F INETIPUX.NT CLEAN.
5. Run NMAKE /F INETIPUX.NT.

Related User Exits

None

RSCUSERENTRY—Client Side RSC/MP Distributed Processing Flow Data Access Exit, Targeting HP NonStop Servers (Windows)

```
int RSCUserEntry(pTRANSHANDLE const transHandle, int sending );
```

Source Code

IORSCCLX.CXX

Purpose

The RSCUserEntry user exit supports multiple APIs that allow inspection and modification of user data and application data before it is sent to the target server. User and application data received from the target server can also be inspected or modified before forwarding to the client application.

Note: The size of the CA Gen data buffer cannot be modified. Data integrity must be maintained by the user exit.

Each user exit API may be called just before sending data to the target server and again just after receiving data from the target server. This enables use of the APIs for such purposes as encrypting/decrypting transaction data, adding custom data that is not applicable to CA Gen to the buffer sent to the server side, and performing customized auditing.

More information:

API Functions (see page 136)

APIs

The following table describes each API supported by the Remote Server Call interface, RSCUserEntry.

API Name	Description
GetMessageSize	Returns the length of the message area for the current request or response message.
GetUserData	Returns the user data associated with the current request or response message.
SetUserData	Copies the passed in buffer into the user data area associated with the current request or response message.
GetIEFData	Returns the CA Gen data (and its length) contained within the current receive or response message.

API Name	Description
SetIEFData	Modifies the CA Gen data contained within the buffer used for the current receive or response message.

Note: For more information about the API functions for the RSCUserEntry user exit, see API Functions (see page 136).

Return Code

This exit should return True if the operation performed succeeds or returns False if the operation fails.

Default Behavior

RSCUserEntry is invoked only when the user exit .dll file is configured using the Client Manager RSC/MP Configuration Details dialog. The sample exit supplied contains code which is commented out, thus no operations are performed.

Building the Exit

Build the RSC/MP user exit to make the functionality provided by the APIs accessible. The Microsoft Visual C++ compiler should be installed on the system where you build the .dll. The exit is built as a part of the dynamic link library, RSCUXxxN.dll; the xx is the CA Gen release number.

Follow these steps:

1. Launch a Command Prompt window.
2. Change to the directory that contains the makefile IORSCUX.NT. (By default, this file is in the CA Gen installation's sample\CommBridge subdirectory.)
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F IORSCUX.NT.
5. Copy the RSCUXxxN.DLL user exit into the Comm Bridge installation directory.

Related User Exits

USEREXIT—The RSC/MP Server Side User Exit

API Functions

This section provides information about how to use the API Functions of the RSCUserEntry user exit.

Function Format

```
int GetMessageSize( pTRANSHANDLE transHandle, short * msgSize )
```

Purpose

The GetMessageSize API function returns the size of the current message buffer. This includes the NonStop RSC/MP header, Gen data, and user data.

If the second parameter passed into the main entry point, RSCUserEntry(), is 1 then this signifies that the exit is being called just prior sending the request message to the target server. The message size returned will be that of the message about to be sent to the server.

If the second parameter passed into the main entry point, RSCUserEntry(), is 0 then this signifies that the exit is being called just after the response message has been received from the target server. The message size returned is that of the message just received from the server.

Arguments

The following table describes the arguments for the GetMessageSize API Function:

Name	I/O	Description
transHandle	Input	The transaction handle for this session. This value is passed into the RSCUserEntry entry point user exit by the CA Gen runtime. This value must not be changed.
msgSize	Output	A pointer to a short where the total size of the current message is stored.

Return Code

The following table describes each return code for the GetMessageSize API function.

Return Code	Description
0	The API function succeeded.
Nonzero	An invalid transHandle was detected.

Default Behavior

When enabled, the sample RSCUserEntry user exit calls this API after receiving a response from the server, prior to forwarding it to the client.

Function Format

```
int GetUserData( pTRANSHANDLE transHandle, unsigned char * data, short * length )
```

Purpose

Returns the user data area associated with the current message. If the second parameter passed into the main entry point, RSCUserEntry(), is 1 then this signifies that the exit is being called just before sending the request message to the target server. The user data returned is the data previously set by the client, if any.

If the second parameter passed into the main entry point, RSCUserEntry(), is 0 then this signifies that the exit is being called just after the response message has been received from the target server. The user data returned is that previously set by the server, if any.

Arguments

The following table describes the arguments for the GetUserData API Function:

Name	I/O	Description
transHandle	Input	The transaction handle for this session. This value is passed into the RSCUserEntry entry point user exit by the CA Gen runtime. This value must not be changed.
* data	Input/Output	A pointer to a buffer area where the runtime is to copy the user data. **Note:** It is the caller's responsibility to ensure adequate memory has been allocated to contain the user data contained in the current message.
* length	Output	A pointer to a short that will contain the actual size of the data copied.

Return Code

The following table describes each return code for the GetUserData API function:

Return Code	Description
0	The API function succeeded.
Nonzero	An invalid transHandle was detected.

Default Behavior

If enabled, the sample RSCUserEntry user exit calls this API after having received a response from the server, before forwarding it to the client.

Function Format

```
int SetUserData( pTRANSHANDLE transHandle, unsigned char * data, short * length )
```

Purpose

Copies the passed in buffer into the user data area associated with the current message.

If the second parameter passed into the main entry point, RSCUserEntry(), is 1 this signifies that the exit is being called just prior to sending the request message to the target server. The buffer data will be copied into the current message's user data area prior to the message being sent to the target server.

If the second parameter passed into the main entry point, RSCUserEntry(), is 0 this signifies that the exit is being called just after the response message has been received from the target server. The user data returned is that previously set by the server, if any.

Arguments

The following table describes the arguments for the SetUserData API Function:

Name	I/O	Description
transHandle	Input	The transaction handle for this session. This value is passed into the RSCUserEntry entry point by the Gen runtime. This value must not be changed.
* data	Output	A pointer to a buffer area that is copied into the current message's user data area.
length	Output	The length of the data buffer pointed to by the data parameter

Return Code

The following table describes each return code for the SetUserData API function:

Return Code	Description
0	The API function succeeded.
Nonzero	An invalid transHandle was detected.

Default Behavior

If enabled, the sample RSCUserEntry user exit calls the SetUserData API prior to the request being sent to the server.

Function Format

```
int GetIEFData( pTRANSHANDLE transHandle, short sending, unsigned char * data, short
* length )
```

Purpose

Returns a copy of the client data associated with the current message.

If the second parameter passed into the main entry point, RSCUserEntry(), is 1 then this signifies the exit is being called just prior sending the request message to the target server. The client data returned will be that which will be forwarded on to the server.

If the second parameter passed into the main entry point, RSCUserEntry(), is 0 then this signifies the exit is being called just after the response message has been received from the target server. The data returned will be that which will be forwarded on to the client.

The following table describes the arguments for the SetUserData API Function:

Name	I/O	Description
transHandle	Input	The transaction handle for this session. This value is passed into the RSCUserEntry entry point by the Gen runtime. This value must not be changed.
Sending	Output	This value must be set the same as the second parameter passed into RSCUserEntry() by the runtime.
* data	Input/Output	A pointer to a buffer area where the runtime is to copy the client data. **Note:** It is the caller's responsibility to ensure that adequate memory has been allocated to contain the client data copied from the current message.
* length	Output	A pointer to a short that will contain the actual size of the data copied.

The following table describes each return code for the SetUserData API function:

Return Code	Description
0	The API function succeeded.
Nonzero	An invalid transHandle was detected.

Default Behavior

If enabled, the sample RSCUserEntry user exit calls the SetUserData API just before sending the request to the server and calls the API again just after the server sends a response.

Function Format

```
int SetIEFData( pTRANSHANDLE transHandle, short sending, unsigned char * data )
```

Purpose

Copies the passed in buffer into the client data area associated with the current message.

If the second parameter passed into the main entry point, RSCUserEntry(), is 1, then this signifies that the user exit is being called just before sending the request message to the target server. The buffer data will be copied into the current message's client data area prior to the message being forwarded to the target server.

If the second parameter passed into the main entry point, RSCUserEntry(), is 0, then this signifies that the user exit is being called just after the response message has been received from the target server. The buffer data will be copied into the current message's client data area prior to the message being forwarded to the client.

Arguments

The following table describes the arguments for the SetUserData API Function:

Name	I/O	Description
transHandle	Input	The transaction handle for this session. This value is passed into the RSCUserEntry entry point by the Gen runtime. This value must not be changed.
Sending	Output	This value must be set the same as the second parameter passed into RSCUserEntry() by the runtime.
* data	Input/Output	A pointer to a buffer area that is copied into the current message's client data area. The number of bytes copied will correspond to the client data size as know by the runtime. It is the caller's responsibility to maintain the integrity of the client data structure and size.

Return Code

The following table gives a brief description of each of the return code values:

Return Code	Description
0	Indicates that the API completed successfully.
Nonzero	Indicates that the API detected an invalid transHandle.

Default Behavior

If enabled, the default user exit calls this API just before sending the request to the server and again just after the response has been returned from the server.

Common System Utilities - Windows User Exits

All supplied Common System Utilities (CSU) user exits are written using the C++ programming language. The following table briefly describes the Common System Utilities Exits:

Common System Utilities: Language: C++		
User Exit Name	**Source Code**	**Description**
CSUGETLIBRARYVERSIONNAME	csuglvn.cxx	Provide a mapping of specified name to version specific name of Library DLL's

Details for the preceding user exits follow in a separate section for each.

CSUGETLIBRARYVERSIONNAME—Version Name mapping Exit

```
void CSUGetLibraryVersionName (char *name,
    char *retName,
    long maxRetNameLen )
```

Source Code

CSUGLVN.CXX

Purpose

This exit provides a mapping of specified name to version specific name if one exists, otherwise the specific name value is returned. This mapping is used when Library DLL's are dynamically loaded during cooperative processing.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*name	Input	Pointer to a character string containing a null terminated name to be converted to a version name.
*retName	Output	Pointer to a character string to receive the version name.
maxRetNameLen	Input	Long containing the maximum length of the returning version name.

Return Code

None

Default Behavior

The default mapping table is used.

Building on Windows

The Windows CSU Library Version Name exit is built as part of the dynamic link library CSUVNxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates the platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CSUGLVN.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CSUGLVN.NT.

Related User Exits

None

TCP/IP - Windows User Exits

All supplied TCP/IP Transport user exits are written using the C programming language. The following table briefly describes the TCP/IP Exits:

TCP/IP Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_TCP_DPC_DIRSERV_EXIT	citcpclx.c	TCP/IP Directory Services User Exit
CI_TCP_DPC_HANDLECOMM_COMPLETE	citcpclx.c	Verifies that a data has been processed successfully (a valid send/receive has occurred).
CI_TCP_DPC_SETUPCOMM_COMPLETE	citcpclx.c	Verifies that connection to target server is successful.

Details for the preceding user exits follow in a separate section for each.

CI_TCP_DPC_DIRSERV_EXIT—TCPIP DPC Directory Services Exit

```
Void CI_TCP_DPC_DirServ_Exit (char *hostName,
    char *servName,
    char *nextLoc,
    char *trancode,
    char *procName,
    char *modelName);
```

Source Code

CITCPCLX.C

Purpose

The provided sample TCPIP DPC Directory Services exit is an implementation of Transaction routing. Transaction routing is a conceptual process that lets cooperative flow data be routed from a Distributed Process Client (DPC) to a programmatically determined Distributed Process Server (DPS). The supplied sample exit looks for the hostname or IP address and port number or service name in environment variables. The user is free to implement whatever functionality can be required.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*hostName	Input/ Output	The hostname where the target server environment resides according to the configured client.
*servName	Input/ Output	The port number or service name on the target server environment the client is to connect to.
*nextLoc	Input	Next Location system attribute as set using CA Gen action diagram statements.
*trancode	Input	The target Procedure Step transaction code being processed
*procName	Input	The name of the flow's target Procedure Step.
*modelName	Input	The name of the model containing the target Procedure Step.

Return Code

None

Default Behavior

If the environment variables expected by the sample implementation of this exit are not defined the hostname and service name values defined during the packaging of the cooperative model will be used.

Building on Windows

The Windows TCP/IP DPC Directory Services exit is built as part of the dynamic link library TCPCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CTCPEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CTCPEXIT.NT.

Related User Exits

None

CI_TCP_DPC_HANDLECOMM_COMPLETE—TCP/IP DPC Handle Comm Complete Exit

```
int CI_TCP_DPC_handleComm_Complete(int completionCode,
    int numberOfAttempts,
    unsigned long reasonCode)
```

Source Code

CITCPCLX.C

Purpose

The processing of a given cooperative flow is broken up into two large grain activities. This, the second is handleComm, which is invoked to send/receive data over an already active connection. This exit is invoked at the completion of the handleComm processing to expose that processing results. The handleComm will either be successful (indicated by the input parameter completionCode having a value of HANDLECOMM_OK) or not successful (indicated by the completionCode parameter having a value of HANDLECOMM_NOT_OK).

The return from this exit indicates if the processing of the cooperative flow should continue (zero lets the process continue, non-zero causes the processing of the flow to be terminated).Therefore, if a completionCode of HANDLECOMM_OK is received as input, the return value from this exit should be set to zero to let the processing of the flow continue.

If the completionCode has a value of HANDLECOMM_NOT_OK, this exit has the opportunity to indicate if the handleComm processing should be attempted by returning a value of zero (0).The number of flow attempts is passed into this exit using the numberOfAttempts parameter. Thus, this exit can control the number of retry attempts by testing the value of numberOfAttempts and returning one (1) when the number of retries has reached a predetermined threshold.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
completionCode	Input	An integer value representing the result of a handleComm operation. The value will be either HANDLECOMM_OK or HANDLECOMM_NOT_OK
numberOfAttempts	Input	An integer value representing the number of times a handleComm operation has been attempted. This number will be incremented each time handleComm fails.
reasonCode	Input	An unsigned long value providing the error number as reported by the underlying TCP/IP transport layer. This error code can be used by this exit to determine if a retry is feasible.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
zero (0)	Indicates processing of the flow should continue, retrying the flow processing if not already successful.
non-zero	Causes the processing of the flow to be terminated.

Default Behavior

The default implementation of this exit lets the flow processing be attempted twice. If the flow is not successful after two attempts, the flow processing is terminated and an appropriate error response is returned to the DP client.

Building on Windows

The Windows TCP/IP DPC Comm Complete exit is built as part of the dynamic link library TCPCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. *N* indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CTCPEXIT.NT (by default, the CA Gen installation directory).
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CTCPEXIT.NT.

Related User Exits

CI_TCP_DPC_SETUPCOMM_COMPLETE

CI_TCP_DPC_SETUPCOMM_COMPLETE—TCP/IP DPC Setup Comm Complete Exit

```
int CI_TCP_DPC_setupComm_Complete (int completionCode,
    int numberOfAttempts,
    unsigned long reasonCode);
```

Source Code

CITCPCLX.C

Purpose

The processing of a given cooperative flow is broken up into two large grain activities. The first is setupComm, which is invoked to insure a connection to the target server is available. This exit is invoked at the completion of the setupComm processing to expose that processing results. The setupComm will either be successful (indicated by the input parameter completionCode having a value of SETUPCOMM_OK) or not successful (indicated by the completionCode parameter having a value of SETUPCOMM_NOT_OK).

The return from this exit indicates if the processing of the cooperative flow should continue (zero lets the process continue, non-zero causes the processing of the flow to be terminated). If the completionCode has a value of SETUPCOMM_NOT_OK, this exit has the opportunity to indicate if the setupComm processing should be attempted by returning a value of zero(0). The number of connection retries attempted is passed into this exit using the numberOfAttempts parameter. Thus, this exit can control the number of retry attempts by testing the value of numberOfAttempts and returning one (1) when the number of retries has reached a predetermined threshold.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
completionCode	Input	An integer value representing the result of a setupComm operation. The value will be either SETUPCOMM_OK or SETUPCOMM_NOT_OK
numberOfAttempts	Input	An integer value representing the number of times a setupComm has been attempted. This number will be incremented each time setupComm fails.
reasonCode	Input	An unsigned long value providing the error number as reported by the underlying TCP/IP transport layer. This error code can be used by this exit to determine if a retry is feasible.

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
zero (0)	Indicates processing of the flow should continue, retrying the connection if not already established.
non-zero	Causes the processing of the flow to be terminated.

Default Behavior

The default implementation of this exit lets the connection be attempted twice. If the connection is not established after two attempts, the flow processing is terminated and an appropriate error response is returned to the DP client.

Building on Windows

The Windows DPC Setup Comm Complete exit is built as part of the dynamic link library TCPCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CTCPEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CTCPEXIT.NT.

Related User Exits

CI_TCP_DPC_HANDLECOMM_COMPLETE

WebSphere MQ Client Transport - Windows User Exits

All supplied WebSphere MQ Client Transport user exits are written using the C programming language. The following table briefly describes the WebSphere MQ Exits:

WebSphere MQ Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_MQS_DPC_EXIT	cimqclex.c	MQ Directory Services Exit
CI_MQS_DPC_HANDLECOMM_COMPLETE	cimqclex.c	Verifies that a data has been processed successfully (a valid send/receive has occurred).
CI_MQS_DPC_SETREPORTOPTIONS	cimqclex.c	Used to override report options set by the runtime.

WebSphere MQ Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_MQS_DPC_SETUPCOMM_COMPLETE	cimqclex.c	Verifies that connection to target server is successful.
CI_MQS_DYNAMICQNAME_EXIT	cimqclex.c	Provide Queue Name that will be used when opening a dynamic queue.
CI_MQS_MQSHUTDOWNTEST	cimqclex.c	Determine if queue should be removed and thus disconnected.

Details for the preceding user exits follow in a separate section for each.

CI_MQS_DPC_EXIT—MQSeries DPC Directory Services Exit

```
Void CI_MQS_DPC_Exit (char *qMgr,
    char *rqMgr,
    char *pQ,
    char *rQ,
    long *timeout,
    short *closePQ,
    short *closeGQ,
    char *nextLoc,
    char *trancode,
    char *procName,
    char *modelName);
```

Source Code

CIMQCLEX.C

Purpose

The provided sample WebSphere MQ DPC Directory Services exit is an implementation of Transaction routing. Transaction routing is a conceptual process that lets cooperative flow data be routed from a Distributed Process Client (DPC) to a programmatically determined Distributed Process Server (DPS). The current cooperative request's local queue manager, remote queue manager, put and reply queue names can be overridden using this exit. Additionally a get timeout value as well as put/get queue disposition after a flow has completed, can be customized.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*qMgr	Input/Output	The name of the local queue manager. By default, the application obtains this information from the model during generation. This exit can override this name.
*rqMgr	Input/Output	The name of the remote queue manager. This value is NULL by default. This exit can override this name.
*pQ	Input/Output	A character string containing the name of the Put queue. By default, the application obtains this information from the model during generation. This exit can override this name.
*rQ	Input/Output	A character string containing the name of the reply-to queue. This can be either a local queue or a model queue name. By default, the application obtains the value "SYSTEM.DEFAULT.MODEL.QUEUE" from the model during generation. This exit can override this name.
*timeout	Input/Output	A long value representing the timeout value, in milliseconds, for the Get queue. By default, this has the value MQWI_UNLIMITED for an unlimited waiting period. This exit can override this name.
*closePQ	Input/Output	A short value which controls whether the client closes the Put queue after the flow is complete. Valid values are CLOSE_QUEUE or NO_CLOSE_QUEUE. The default value, NO_CLOSE_QUEUE, specifies the Put queue is not to be closed. This exit can override this name.
*closeGQ	Input/Output	A short value which controls whether the client closes the Get queue after the flow is complete. Valid values are CLOSE_QUEUE or NO_CLOSE_QUEUE. The default value, NO_CLOSE_QUEUE, specifies the Get queue is not to be closed. This exit can override this name.
*nextLoc	Input	A character string containing the Next Location system attribute as set using CA Gen action diagram statements.
*trancode	Input	An 8-byte character array containing the target Procedure Step transaction code being processed.
*procName	Input	A character string containing the name of the flow's target Procedure Step.

Name	I/O	Description
*modelName	Input	A character string containing the name of the model containing the flow's target Procedure Step.

Return Code

None

Default Behavior

The supplied sample does not implement dynamic transaction routing. For more information about default values for the various parameters, see Arguments.

Building on Windows

The Windows WebSphere MQ DPC Directory Services exit is built as part of the dynamic link library MQSCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CMQSEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CMQSEXIT.NT.

Related User Exits

None

CI_MQS_DPC_HANDLECOMM_COMPLETE—Handle Comm Retry Count Exit

```
int CI_MQS_DPC_handleComm_Complete(int completionCode,
    int numberOfAttempts,
    unsigned long reasonCode);
```

Source Code

CIMQCLEX.C

Purpose

The processing of a given cooperative flow is broken up into two large grain activities. The second is handleComm, which is invoked to send/receive data over an already active connection. This exit is invoked at the completion of the handleComm processing to expose that processing results. The handleComm will either be successful (indicated by the input parameter completionCode having a value of HANDLECOMM_OK) or not successful (indicated by the completionCode parameter having a value of HANDLECOMM_NOT_OK).

The return from this exit indicates if the processing of the cooperative flow should continue (zero lets the process continue, non-zero causes the processing of the flow to be terminated). Therefore, if a completion code of HANDLECOMM_OK is received as input, the return value from this exit should be set to zero to let the processing of the flow continue.

If the completionCode has a value of HANDLECOMM_NOT_OK, this exit has the opportunity to indicate if the handleComm processing should be attempted by returning a value of zero (0). The number of flow attempts is passed into this exit using the numberOfAttempts parameter. Thus, this exit can control the number of retry attempts by testing the value of numberOfAttempts and returning one (1) when the number of retries has reached a predetermined threshold.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
completionCode	Input	An integer value representing the result of a handleComm operation. The value will be either HANDLECOMM_OK or HANDLECOMM_NOT_OK
numberOfAttempts	Input	An integer value representing the number of times a handleComm operation has been attempted. This number will be incremented each time handleComm fails.
reasonCode	Input	An unsigned long value providing the error number as reported by the underlying WebSphere MQ transport layer. This error code can be used by this exit to determine if a retry is feasible.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
zero (0)	Indicates processing of the flow should continue, retrying the flow processing if not already successful.
non-zero	Causes the processing of the flow to be terminated.

Default Behavior

The default implementation of this exit lets the flow processing be attempted twice. If the flow is not successful after two attempts, the flow processing is terminated and an appropriate error response is returned to the DP client.

Building on Windows

The Windows WebSphere MQ Handle Comm Retry Count exit is built as part of the dynamic link library MQSCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CMQSEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CMQSEXIT.NT.

Related User Exits

CI_MQS_DPC_SETUPCOMM_COMPLETE

CI_MQS_DPC_SETREPORTOPTIONS—Override Put Queue Report Options Exit Description

```
void CI_MQS_DPC_setReportOptions( MQLONG * reportOptions);
```

Source Code

CIMQCLEX.C

Purpose

This exit can be used to override the set of report options defined for the WebSphere MQ Put Message Descriptor prior to the issuance of an MQPUT() operation.

The report options set by the runtime are described in the following table:

Report Option	Description
MQRO_EXCEPTION	This type of report can be generated when an exception occurs. For instance if a message is sent to another queue manager and the message cannot be delivered to the specified destination queue.
MQRO_EXPIRATION	An expiration report. The queue manager generates this type of report if the message is discarded prior to delivery to an application because its expiry time has passed.
MQRO_PASS_MSG_ID	If a report is generated, the MsgId of the current message being processed is to be copied to the MsgId of the report message.
MQRO_COPY_MSG_ID_TO_CORREL_ID	Indicates the correlation ID of the report generated should equal the message ID of the request originally issued.
MQRO_DEAD_LETTER_Q	This option causes the original message to be placed on the dead-letter queue when an exception occurs

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*reportOptions	Input/ Output	A pointer to a long value representing the currently defined report options to be used in the Put Message Descriptor.

Return Code

None

Default Behavior

The runtime specified report options are left unchanged.

Building on Windows

The Windows WebSphere MQ Override Put Queue Report Options exit is built as part of the dynamic link library MQSCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CMQSEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CMQSEXIT.NT.

Related User Exits

None

CI_MQS_DPC_SETUPCOMM_COMPLETE—Setup Comm Retry Count Exit

```
int CI_MQS_DPC_setupComm_Complete(int completionCode,
    int numberOfAttempts,
    unsigned long reasonCode);
```

Source Code

CIMQCLEX.C

Purpose

The processing of a given cooperative flow is broken up into two large grain activities. The first is setupComm, which is invoked to insure a connection to the target server is available. This exit is invoked at the completion of the setupComm processing to expose that processing results. The setupComm will either be successful (indicated by the input parameter completionCode having a value of SETUPCOMM_OK) or not successful (indicated by the completionCode parameter having a value of SETUPCOMM_NOT_OK).

The return from this exit indicates if the processing of the cooperative flow should continue (zero lets the process continue, non-zero causes the processing of the flow to be terminated). If the completionCode has a value of SETUPCOMM_NOT_OK, this exit has the opportunity to indicate if the setupComm processing should be attempted by returning a value of zero (0). The number of connection retries attempted is passed into this exit using the numberOfAttempts parameter. Thus, this exit can control the number of retry attempts by testing the value of numberOfAttempts and returning one (1) when the number of retries has reached a predetermined threshold.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
completionCode	Input	An integer value representing the result of a setupComm operation. The value will be either SETUPCOMM_OK or SETUPCOMM_NOT_OK.
numberOfAttempts	Input	An integer value representing the number of times a setupComm has been attempted. This number will be incremented each time setupComm fails.
reasonCode	Input	An unsigned long value providing the error number as reported by the underlying WebSphere MQ transport layer. This error code can be used by this exit to determine if a retry is feasible.

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
zero (0)	Indicates processing of the flow should continue, retrying the connection if not already established.
non-zero	Causes the processing of the flow to be terminated.

Default Behavior

The default implementation of this exit lets the connection be attempted twice. If the connection is not established after two attempts, the flow processing is terminated and an appropriate error response is returned to the DP client.

Building on Windows

The Windows WebSphere MQ Setup Comm Retry Count exit is built as part of the dynamic link library MQSCXnnN.DLL, where *xx* is the CA Gen release number and *N* indicates platform. A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CMQSEXIT.NT (by default, the CA Gen installation directory).
3. Set Microsoft Visual Studio compiler environment variables.
4. Run nmake /f CMQSEXIT.NT.

Related User Exits

CI_MQS_DPC_HANDLECOMM_COMPLETE

CI_MQS_DYNAMICQNAME_EXIT—Dynamic Queue Name Override Exit

```
void CI_MQS_DynamicQName_Exit (char *dynamicQName);
```

Source Code

CIMQCLEX.C

Purpose

The Dynamic Queue Name exit lets you override the queue name that will be used when opening a dynamic queue. The resulting Dynamic Queue will obtain its attributes from the specified WebSphere MQ Model Queue name. The passed in work area can be modified by placing a null terminated string of the value to be used as the dynamic queue name, including the use of valid WebSphere MQ pattern characters used to name dynamic queues.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*dynamicQName	Input/ Output	A pointer to a character buffer, of length MQ_Q_NAME_LENGTH+1 (48 +1), that contains the default name of the dynamic queue as built by the WebSphere MQ runtime (for example username.processid.threadid).

Return Code

None

Default Behavior

The dynamic queue name is not modified.

Building on Windows

The Windows WebSphere MQ Dynamic Queue Name Override exit is built as part of the dynamic link library MQSCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CMQSEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CMQSEXIT.NT.

Related User Exits

None

CI_MQS_MQSHUTDOWNTEST—MQSeries Queue Disconnect Exit

```
Int CI_MQS_MQShutdownTest()
```

Source Code

CIMQCLEX.C

Purpose

This exit can be used to modify the behavior of the normal put/get queue disposition after successful completion of a cooperative flow. Normal disposition will leave the connection valid with the put and get queues open, ready to handle subsequent flows. This exit can override that behavior and cause the queues and connection to be closed after each flow has completed.

Arguments

None

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
NO_REMOVE_QUEUE	The default return value. The connection and put/get queues will remain open, available for subsequent flows.
REMOVE_QUEUE	After a completed flow the connection is dropped, the put/get queues will be closed.

Default Behavior

A value of NO_REMOVE_QUEUE is returned, the connection and put/get queues remain open and available for subsequent flows.

Building on Windows

The Windows WebSphere MQ Queue Disconnect exit is built as part of the dynamic link library MQSCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. Nindicates platform.

A prerequisite for building the DLL, you must have Microsoft Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CMQSEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CMQSEXIT.NT.

Related User Exits

None

ECI - Windows User Exits

All supplied ECI Transport user exits are written using the C programming language. The following table briefly describes the ECI Exits:

ECI Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_ECI_GET_SYSTEM_NAME	cieciclx.c	Provide name of the target CICS System
CI_ECI_GET_TPN	cieciclx.c	Provide the specified CICS Mirror Transaction associated with the request

Details for the preceding user exits follow in a separate section for each.

CI_ECI_GET_SYSTEM_NAME—Get ECI System Name Exit (Windows)

```
short ci_eci_get_system_name ( char * pTranCode,
    char * pNextLoc,
    char * pProcName,
    char * pModelName,
    char * pSysName );
```

Source Code

CIECICLX.C

Purpose

The Get ECI System Name exit is an implementation of Transaction routing for ECI. Transaction routing is a conceptual process that lets cooperative flow data be routed from a Distributed Process Client (DPC) to a programmatically determined Distributed Process Server (DPS). This exit is called from the ECI cooperative flow runtime, prior to invoking the ci_eci_get_tpn() exit and prior to invoking CICS_ExternalCall() API. This exit is called to obtain the name of the target CICS System (as it is known to the CICS Universal Client).

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*pTranCode	Input	A pointer to an 8-character array containing the transaction code to be invoked using the ECI call.

Name	I/O	Description
*pNextLoc	Input	A pointer to a character array containing the Next Location system attribute as set within the DP client using CA Gen action diagram statements.
*pProcName	Input	A pointer to a character array containing the target Procedure Step name.
*pModelName	Input	A pointer to a character array containing the model name containing the target Procedure Step.
*pSysName	Input/Output	A Pointer to a character array on input that can contain a CICS System name if previously supplied by the commcfg.ini file. The memory area pointed to by this address contains the name of the CICS system to be used on the ECI API call. This exit can override the passed in system name. The character array is defined to receive a null terminated string of max length CICS_ECI_SYSTEM_MAX+1 (8 +1).

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
zero (0)	If the character array pointed to by the pSysName has been populated with a CICS system name.
non-zero	If the CICS system name is undetermined and will result in the eci_system_name to not be populated when the ECI CoopFlow issues the CICS_ExternalCall().

Default Behavior

The default operation of this exit will, if the system name is populated on input, return a 0 (indicating the supplied CICS system name is to be used). If the system name is not populated on input, the exit returns the default system name, as is returned by the CICS_EciListSystems() call (assuming the CICS_EciListSystems() call returns ECI_NO_ERROR).

Building on Windows

The Windows ECI Get ECI System Name exit is built as part of the dynamic link library ECICXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. *N* indicates platform.

A prerequisite for building the DLL, you must have Microsoft Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CECIEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CECIEXIT.NT.

Related User Exits

CI_ECI_GET_TPN—Get ECI Mirror Transaction Exit

CI_ECI_GET_TPN—Get ECI Mirror Transaction Exit (Windows)

```
void ci_eci_get_tpn ( char * pTranCode,
    char * pNextLoc,
    char * pProcName,
    char * pModelName,
    char * pSysName,
    char * pTpn );
```

Source Code

CIECICLX.C

Purpose

The Get ECI Mirror Transaction exit lets you override the name of the CICS ECI Mirror Transaction used for the current request. Several request identifying data items are passed into the exit to allow flow-by-flow customization of the Mirror Transaction. This exit is called from the ECI cooperative flow runtime, after invoking the ci_eci_get_system_name() exit and prior to invoking CICS_ExternalCall() API.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*pTranCode	Input	A pointer to an 8-character array containing the transaction code to be invoked using the ECI call.
*pNextLoc	Input	A pointer to a character array containing the Next Location system attribute as set within the DP client using CA Gen action diagram statements.
*pProcName	Input	A pointer to a character array containing the target Procedure Step name.
*pModelName	Input	A pointer to a character array containing the model name containing the target Procedure Step.
*pSysName	Input	A Pointer to a character array containing the CICS System name which is the target of the request. This value can have been previously customized using the ci_eci_get_system_name user exit.
*pTpn	Input/Output	A pointer to a four-character array containing the CICS ECI Mirror Transaction name. This exit by default will set this argument to CPMI. If the array is set to contain NULLs, CPMI will also be used for the Mirror Transaction.

Return Code

None

Default Behavior

The default operation of this exit will set the name of the Mirror Transaction to CPMI. If the array pointed to by pTpn is set to NULLs, the Mirror Transaction will also default to CPMI.

Building on Windows

The Windows ECI Get ECI System Name exit is built as part of the dynamic link library ECICXxxN.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. *N* indicates platform.

A prerequisite for building the DLL, you must have Microsoft Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CECIEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run nmake /f CECIEXIT.NT.

Related User Exits

CI_ECI_GET_SYSTEM_NAME—Get ECI System Name Exit

Tuxedo

All supplied Tuxedo Transport user exits are written using the C programming language. The following table briefly describes the Tuxedo Exits:

Tuxedo Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_C_SEC_SET	cictuxwsx.c	Set User Supplied Security Data Into The Security Data Fields Located In Tuxedo Tpinit
CI_C_USER_DATA_IN	cictuxwsx.c	Gives You The Opportunity To Inspect Or Modify The Cooperative Flow Request Buffer On Return From The Target Tuxedo Service. Invoked On Return From The Tpcall For Those Clients Connecting To Tuxedo Servers Residing On A Separate Host. Additionally, This Exit Allows The Client To Disconnect From The Server Following Each Flow

Tuxedo Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_C_USER_DATA_IN	cictuxx.c	Gives You The Opportunity To Inspect Or Modify The Cooperative Flow Request Buffer On Return From The Target Tuxedo Service. Invoked On Return From The Tpcall For Those Clients Connecting To Tuxedo Servers Residing On A Separate Host. Additionally, This Exit Allows The Client To Disconnect From The Server Following Each Flow (used For Server To Server Flows)
CI_C_USER_DATA_OUT	cictuxwsx.c	Gives You The Opportunity To Inspect Or Modify The Cooperative Flow Request Buffer Prior To Tuxedo Sending The Request To The Target Tuxedo Service. Invoked Prior To The Tpcall For Those Clients Connecting To Tuxedo Servers Residing On A Separate Host
CI_C_USER_DATA_OUT	cictuxx.c	Gives You The Opportunity To Inspect Or Modify The Cooperative Flow Request Buffer Prior To Tuxedo Sending The Request To The Target Tuxedo Service. Invoked Prior To The Tpcall For Those Clients Connecting To Tuxedo Servers Residing On A Separate Host (used For Server To Server Flows)
CI_EVENT_HANDLER	cictuxwsx.c	Provides ability to handle events

Note that both cictuxwsx.c and cictuxx.c have essentially the same user exits. The difference is that the user exits in cictuxwsx.c are used for client/server flows, while the user exits in cictuxx.c are used for server to server flows.

Details for the preceding user exits follow in a separate section for each.

CI_C_SEC_SET—Tuxedo Cooperative Flow Security Exit

```
int ci_c_sec_set (CIPROCSTEP *procstep,
    char *clientUserid,
    char *clientPassword,
    int *tperr )
```

Source Code

CICTUXWSX.C

Purpose

The ci_c_sec_set exit is the first user exit that is invoked on a client Tuxedo cooperative flow, and is specifically related to security. ci_c_sec_set() contains the call to tpinit() and is called for each flow so that, if required, each flow can establish a different user context (that is, tpinit() is invoked for each flow). You need to invoke the corresponding tpterm in ci_c_user_data_in() to achieve the described behavior.

ci_c_sec_set() is called from a Windows Client only. A Tuxedo server-to-server flow does not invoke ci_c_sec_set().

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*procstep	Input	A pointer to a CA Gen CIPROCSTEP structure, which contains information, related to the currently executing procedure step. Model name and Next Location data can be extracted from this structure. This parameter is unused in the supplied sample exit.
*clientUserid	Input	CA Gen current CLIENT_USER_ID system attribute, if SecurityUsedEnhanced is set in the WRSECTOKEN user exit. NULL string otherwise.
*clientPassword	Input	CA Gen current CLIENT_PASSWORD system attribute if SecurityUsedEnhanced is set in the WRSECTOKEN user exit. NULL string otherwise.
*tperr	Output	If the function returns -1 as the return code, tperr should contain a valid tperrno value indicating the cause of the Tuxedo ATMI call failure.

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
zero (0)	If no error was encountered within the exit.
non-zero	If one of the ATMI functions within user, the exit fails.

Default Behavior

The clientUserId and clientPassword are used for user name and the data field of the TPINIT structure respectively. TPINIT structure is used for the Tuxedo login. If SecurityUsedEnhanced is returned from the WRSECTOKEN user exit, clientUserid and clientPassword contain a pointer to a valid CA Gen CLIENT_USER_ID and CLIENT_PASSWORD system attributes, which are used for the Tuxedo login. Otherwise, the pointers point to a Null string.

Building on Windows

The client Windows Tuxedo Cooperative Flow Security exit is built as part of the dynamic link library TXWCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CTUXEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CTUXEXIT.NT.

Related User Exits

CI_EVENT_HANDLER

CI_C_USER_DATA_IN—Tuxedo Inbound Flow Data Access Exit

```
void  ci_c_user_data_in(char ** tuxSvcOutputBuffer)
```

Source Code

CICTUXWSX.C, CICTUXX.C

Purpose

The ci_c_user_data_in exit is called immediately after the Tuxedo tpcall API returns. It provides access to the inbound View32 buffer on returning from the target server. tuxSvcOutputBuffer points to the reply buffer, which contains a CA Gen procedure, step structure and export view (back to back).

If required, a call to the Tuxedp tpterm API can be added inside ci_c_user_data_in to force association for each flow with a discrete user environment and privileges (see ci_c_sec_set).

ci_c_user_data_in() is called from both a Windows Client and Server to Server flows, hence the two source files.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
**tuxSvcOutputBuffer	Input	A pointer to a buffer containing a Tuxedo allocated buffer, which contains procedure step and view data for the flow in progress. This is the buffer returned back from the Tuxedo tpcall() API (inbound View32).

Return Code

None

Default Behavior

The default behavior is to not modify the received data.

Building on Windows

The client Windows Tuxedo Inbound Flow Data Access exit is built as part of the dynamic link library TXWCXxxN.DLL. as well as the dynamic link library TXCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CTUXEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CTUXEXIT.NT.

Related User Exits

CI_C_USER_DATA_OUT—Tuxedo Outbound Flow Data Access Exit

CI_C_USER_DATA_OUT—Tuxedo Outbound Flow Data Access Exit

```
void  ci_c_user_data_out(char ** tuxSvcInputBuffer , long * svcFlags );
```

Source Code

CICTUXWSX.C, CICTUXX.C

Purpose

The ci_c_user_data_out() function is the second user exit that is invoked on a cooperative flow. It provides access to the outbound View32 buffer, and an opportunity to add more attributes to the flags parameter of the subsequent Tuxedo tpcall API. tuxSvcInputBuffer is a request buffer containing a CA Gen procedure step structure followed by its import views as back-to-back data items. svcFlags are added to the tpcall flags parameter. Both are passed to the tpcall.

ci_c_user_data_out() is called from both a Windows Client and Server to Server flows, hence the two source files.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
**tuxSvcInputBuffer	Input/ Output	A pointer to a buffer containing a Tuxedo allocated buffer, which contains procedure step and view data for the flow in progress.
*svcFlags	Output	A pointer to a long value to which can be stored extra flags to be appended to the standard value passed as the flags parameter to the Tuxedo tpcall() API.

Return Code

None

Default Behavior

The flow data and flags parameter passed to tpcall() are not modified.

Building on Windows

The client Windows Tuxedo Outbound Flow Data Access exit is built as part of the dynamic link library TXWCXxxN.DLL, as well as the dynamic link library TXCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CTUXEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CTUXEXIT.NT.

Related User Exits

CI_C_USER_DATA_IN—Tuxedo Inbound Flow Data Access Exit

CI_EVENT_HANDLER—Tuxedo Event Handler Exit

void TUXCALL ci_event_handler(char * s, long len, long flag);

Source Code

CICTUXWSX.C

Purpose

The ci_event_handler() function is called in ci_c_sec_set() to respond to the runtime failure from the call to TPINIT().

ci_event_handler() is called from a Windows Client only. A Tuxedo server-to-server flow does not invoke ci_c_sec_set() and therefore does not invoke ci_event_handler().

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*s	Input	A pointer to a buffer containing a Tuxedo error message.
Len	Input	A long value containing the length of the error message.
Flag	Input	A long value containing an error flag.

Return Code

None

Default Behavior

CI_EVENT_HANDLER does no work.

Building on Windows

The client Windows Tuxedo event handler exit is built as part of the dynamic link library TXWCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CTUXEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CTUXEXIT.NT.

Related User Exits

CI_C_SEC_SET

Web Services - Windows User Exits

All supplied Web Services Middleware user exits are written using the C programming language. The following table briefly describes the Web Services Exits:

Web Services Middleware: Language: C		
User Exit Name	**Source Code**	**Description**
CI_WS_DPC_Exit	ciwsclx.c	Programmatic runtime override of parameters (base URL and context type) for Web Service destination.
CI_WS_DPC_URL_Exit	ciwsclx.c	Programmatic runtime override of URL for Web Service destination.

Details for the preceding user exits follow in a separate section for each.

CI_WS_DPC_Exit—Web Services DPC User Exit (Windows)

```
void CI_WS_DPC_Exit(char *baseURL,
    size_t baseURLMaxLen,
    char *contextType,
    size_t contextTypeMaxLen,
    char *modelName,
    char *modelShortName,
    char *trancode,
    char *trancodeAlt,
    char *procName,
    char *procNameAlt,
    char *nextLoc)
```

Purpose

This exit will be called from the C Web Services CoopFlow prior to performing a Web Service connection. It gives the user an opportunity to modify the Web Service endpoint destination by overriding the base URL and the context type.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*baseURL	Input/Output	Scheme, Domain and Port of a Web Service end point URL
baseURLMaxLen	Input	Maximum length of baseURL
*contextType	Input/Output	Part of the path of a CA Gen Web Service end point URL
contextTypeMaxLen	Input	Maximum length of contextType
*modelName	Input	Name of the model containing the target Procedure Step
*modelShortName	Input	Short name of the model containing the target Procedure Step
*tranCode	Input	Transaction code of the target Procedure Step being processed
*tranCodeAlt	Input	Alternative name for the transaction code of the target Procedure Step being processed
*procName	Input	Name of the target Procedure Step to be called
*procNameAlt	Input	Alternative name of the target Procedure Step to be called
*nextLoc	Input	Next Location system attribute as set using CA Gen action diagram statements

Return Code

None

Default Behavior

If the base URL and the context Type variables expected by the sample implementation of this exit are not defined, the default values that are defined during the packaging of the cooperative model or overrides from commcfg.ini are used.

Building on Windows

The Windows Web Services DPC user exit is built as part of the dynamic link library WSCX*xx*N.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CWSEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CWSEXIT.NT.

Related User Exits

None

CI_WS_DPC_URL_Exit — Web Services DPC URL User Exit (Windows)

```
void CI_WS_DPC_URL_Exit(char *url,
    size_t urlMaxLen,
    char *modelName,
    char *modelShortName,
    char *trancode,
    char *trancodeAlt,
    char *procName,
    char *procNameAlt,
    char *nextLoc)
```

Purpose

This exit will be called from the C Web Services CoopFlow prior to performing a Web Service connection. It gives the user an opportunity to modify the Web Service endpoint destination URL.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*url	Input/Output	Web Service end point URL
urlMaxLen	Input	Maximum length of Web Service end point URL
*modelName	Input	Name of the model containing the target Procedure Step
*modelShortName	Input	Short name of the model containing the target Procedure Step
*tranCode	Input	Transaction code of the target Procedure Step being processed
*tranCodeAlt	Input	Alternative name for the transaction code of the target Procedure Step being processed
*procName	Input	Name of the target Procedure Step to be called
*procNameAlt	Input	Alternative name of the target Procedure Step to be called
*nextLoc	Input	Next Location system attribute as set using CA Gen action diagram statements

Return Code

None

Default Behavior

If the URL value is not modified in this user exit, its value prior to calling this exit is used.

Building on Windows

The Windows Web Services DPC User exit is built as part of the dynamic link library WSCXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile CWSEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F CWSEXIT.NT.

Related User Exits

None

Windows Servers User Exits

The following table summarizes the functions available through the user exits for generated server applications:

Name	Description
DBCOMMIT	Database commit User Exit. There is one user exit routine for each supported database: ODBC, Oracle, and DB2.
DBCONNCT	Database connection User Exit. There is one user exit routine for each supported database: ODBC, Oracle, and DB2.
DBDISCNT	Database disconnect User Exit. There is one user exit routine for each supported database: ODBC, Oracle, and DB2.

Name	Description
SRVRERROR	Server to Server Error User Exit (Server Only)
TIRDCRYP	Decrypt User Exit (Server Only)
TIRDLCT	Dialect User Exit
TIRDRTL	Default Retry Limit User Exit
TIRELOG	Server Error Logging User Exit (Server Only)
TIRHELP	Help Interface User Exit
TIRMTQB	Message Table User Exit
TIRNCRYP	Encrypt User Exit (Server Only)
TIRSECR	Security Interface User Exit
TIRSECV	Server Security Validation User Exit (Server Only)
TIRSYSID	System ID User Exit
TIRTERMA	User Termination User Exit
TIRTIAR	Database error message User Exit. There is one user exit routine for each supported database: ODBC, Oracle and DB2.
TIRUPDB	MBCS Uppercase Translation User Exit
TIRUPPR	Uppercase Translation User Exit
TIRURTL	Ultimate Retry Limit User Exit
TIRUSRID	User ID User Exit
TIRXINFO	Locale Information User Exit (Server Only)
TIRXLAT	National Language Translation User Exit (Server Only)
TIRYYX	Date User Exit

Note: The database user exits DBCONNCT, DBCOMMIT, DBDISCNT and TIRTIAR are rebuilt into individual DLL's (AECDB2xxN.DLL, AECODBxxN.DLL, AECORAxxN.DLL) using the command procedure mkdbs.bat in %GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit.

Server runtime user exits are rebuilt into the DLL AEUEXITxxN.DLL using the command procedure mkexits.bat found in %GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit. This is the same DLL that is used with Blockmode applications.

Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

Because a large number of these user exits have already been documented in the section Windows Blockmode User Exits, only the user exits that are specific to server applications will be detailed in the following subsections.

SRVRERROR—Server to Server Error Exit (Windows)

```
int SRVRERROR ( char * from,
    char * to,
    char * errLst,
    int dtp,
    int failureType,
    char * failureCommand,
    ErrorToken errorToken);
```

Source Code

TIRSERRX.C

Purpose

This exit is invoked by the calling server when errors occur at the destination server, during a server-to-server flow. This exit can influence the default runtime error behavior in how the detected error is handled. When the NOTPROPAGATE_ERR is returned, the calling procedure step continues the execution, ignoring the fact that an error has occurred in the destination procedure step. When PROPAGATE_ERR is returned, an error message is created and then returned to the calling procedure step.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*from	Input	A character string pointing to the name of the source or calling procedure step
*to	Input	A character string pointing to the name of the target or called procedure step
*errLst	Input	A character string pointing to the destination server's XFAIL message. This message buffer can contain multiple new line terminated strings. The initial portion of the buffer is formatted to fit a 24 line by 80-character screen format. Additional free form data can follow the 24 x 80 lines, up to the maximum of 2048 bytes.
dtp	Input	An integer value of 1 if the *to* procedure step is a distributed transaction participant, 0 if otherwise

Name	I/O	Description
failureType	Input	An enumerated value representing a failure code CFBUILD = 0 - implies the calling procedure step failed to build/parse the message bound for the destination procedure step XFAIL = 1 - implies the destination procedure step execution failed XERR = 2 - implies a communication error between the calling and destination procedure steps
*failureCommand	Input/Output	A command that the destination procedure step can return to the calling procedure step. A maximum of 8 chars plus NULL.
errorToken	Input/Output	This parameter is only used with XFAL messages. errorToken can contain a token constructed by the Error Logging exit (TIRELOG) at the target or called procedure step. (4096 +1 bytes)

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
0 - PROPAGATE_ERR	The error is processed normally within the calling procedure step.
1-NOTPROPAGATE_ERR	The calling procedure step continues the execution ignoring the fact that error occurs in the destination procedure step.

Default Behavior

Errors are propagated to the calling procedure step. An error message is created and returned to the calling procedure step.

Building on Windows

The Server to Server Error User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

None

TIRDCRYP—Server Decryption Exit (Windows)

```
void TIRDCRYP( unsigned char * rp1,
    unsigned char * rp2,
    TIRDCRYP_cmcb * pTIRDCRYP_cmcb);
```

Source Code

TIRDCRYP.C

Purpose

TIRDCRYP is called by the Server Manager after it detects that the client has sent an encrypted cooperative buffer. The Server Manager constructs a work buffer containing the concatenated View Data and Client Security sections. The user is responsible for decrypting the area pointed to by pDataBuffer for IBufferSize bytes.

The inputs pDataBuffer and IDecryptMaxSize as well as the outputs IBufferSize return_code and failure_msg are fields within a structure pointed to by the pTIRDCRYP_cmcb parameter.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*pTIRDCRYP_cmcb	Input/Output	A pointer to a structure containing the following items:
lDecryptMaxSize	Input	A long field that contains the maximum available buffer space (in bytes) that the decrypted data can occupy.
lBufferSize	Input/Output	On input, IBufferSize is the current buffer space (in bytes) of the encrypted data. On output, IBuffferSize should be updated by this exit to contain the length of the decrypted data. The length of the decrypted result cannot exceed lDecryptMaxSize.
*pDataBuffer	Input/Output	On input, a pointer to the starting location of the encrypted View Data and Client Security sections within the CFB work buffer. On output, this exit should ensure this same data area contains the unencrypted versions of the input data. The length of this decrypted result cannot exceed lDecryptMaxSize.
return_code	Output	A two-character array returning the results of the decryption attempt. The following values are supported: DECRYPTION_USED—defined as " " DECRYPTION_SIZE_EXCEEDED_MAX—defined as "01" DECRYPTION_NOT_USED—defined as "02" DECRYPTION_APPLICATION_ERROR—defined as "03"
*failureMsg	Output	The pointer to an 80-character array, to be populated by the exit that can receive a null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of DECRYPTION_APPLICATION_ERROR.

Return Code

None directly, see the preceding return_code (see page 182) structure member.

Default Behavior

Decryption of the data buffer is not attempted.

Building on Windows

The Server Decryption User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

The following are related user exits:

- TIRNCRYP
- WRSECENCRYPT

TIRELOG—Server Error Logging and Error Token Creation Exit (Windows)

```
void TIRELOG(char * rp1,
    char * rp2,
    TIRELOG_CMCB * pTIRELOG_cmcb));
```

Source Code

TIRELOG.C

Purpose

This exit serves two purposes:

- Error logging at the server

- Creation of an error token for transmitting to the client

This exit is called by the server to handle server errors that are encountered during the execution of a distributed processing server that cannot be handled by the runtime or generated code, and normally result in the termination of the application. For example, prior to the execution of a server procedure step, the server extracts view data from the client message and places it in the target procedure step's view. If this extraction fails because of a mismatch between the client definition and the server definition an error response message is created and returned to the client.

The default implementation of this exit returns to the caller without logging the error. It is up to the developer of this user exit to determine what information should be logged and how it should be logged. Some users can choose to log only certain errors; others can choose to log all errors. On some systems, the log can be implemented as a file. To log a server error, simply format the information you wish to log and write it to a file. On other systems, the log can be implemented using system-specific features such as a CICS temporary storage queue (TSQ) as found on z/OS.

To create an error token, move text data to the area pointed to by the elog_error_token member of the TIRELOG_CMCB structure passed into this exit. The error token area is 4097 bytes and must be null-terminated. The error token, which goes through codepage translation when it is transmitted to the client, can be used on the client to customize how the error is handled.

For example, you can modify this exit to return an error token of "RETRY" whenever a certain database contention error occurs. This error token is passed to the client error-handling exit (WRSRVRERROR or WRASYNCSRVRERROR), which makes the final decision on how to handle the error. You can modify the client error-handling exit to reinvoke the flow or USE whenever the error token is "RETRY." This server error-logging exit is called after the error response message is created but before it is transmitted to the client.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only
*pTIRELOG_cmcb	Input/Output	A pointer to a TIRELOG_CMCB structure containing the following items:

Name	I/O	Description
elog_fail_type	Input	A character designating the type of failure detected defined as:(lable - defined value) EPROFD - 'P' profile error EPROFI - 'I' profile error EEXEC - 'E' execution error ESERVER- 'D' server manager error EUSER 'U' user requested abend
void *elog_sqlca	Input	A pointer to a saved sql data area
*elog_globdata	Input	A pointer to the server's globdata area
elog_number_of_lines	Input	An integer containing the number of text lines contained within the elog_error_text buffer
elog_error_text	Input	A pointer to a buffer of screen formatted text. This data, formatted by the server runtime, contains up to 24 lines of 80 characters each.
*elog_error_token	Input/Output	A character pointer to an error token area that can contain up to 4097 bytes, this includes the required null terminator. This exit is responsible for populating this data area if needed.

Return Code

None directly, see the preceding pTIRELOG_cmcb structure.

Default Behavior

The default action is to return without logging the error.

Building on Windows

The Server Error Logging User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

The following are related user exits:

- WRSRVRERROR
- WRASYNCSRVRERROR

TIRNCRYP—Server Encryption Exit (Windows)

```
void TIRNCRYP( unsigned char * rp1,
    unsigned char * rp2,
    TIRNCRYP_cmcb * pTIRNCRYP_cmcb);
```

Source Code

TIRNCRYP.C

Purpose

After a server procedure step executes, the server manager can call TIRNCRYP to encrypt the server response to the client. The server manager makes a copy of the unencrypted cooperative buffer pending transmission back to the client. The inputs pDataBuffer, lBufferSize, lEncryptMaxSize trancode and client_userid as well as the outputs return_code, and failure_msg are fields with a structure pointed to by pTIRNCRYP_cmcb. The user is responsible for encrypting the data area pointed to by the pDataBuffer member of the TIRNCRYP_cmcb structure.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*pTIRNCRYP_cmcb	Input/Output	A pointer to a structure containing the following items:
pDataBuffer	Input/Output	On input, a pointer to the starting location of the View Data and Client Security sections within the CFB work buffer. On output this same data area should be populate by this exit with the encrypted versions of the input data. The length of the encrypted result cannot exceed lEncryptMaxSize.
lBufferSize	Input/Output	On input, lBufferSize is the current buffer space (in bytes) of the unencrypted data. On output, lBuffferSize should be updated by this exit to contain the length of the encrypted data. The length of the encrypted result cannot exceed lEncryptMaxSize.
lEncryptMaxSize	Input	A long field that contains the maximum available buffer space (in bytes) that the encrypted data can occupy.
trancode	Input	Transaction code currently being processed. . This value can be used in conjunction with client userid and NextLocation to determine if encryption is desired.
client_userid	Input	Client user ID. This value can be used in conjunction with trancode and NextLocation to determine if encryption is desired.

Name	I/O	Description
pNextLocation	Input	Next Location value as set by the server application using CA Gen action diagram statements. This value can be used in conjunction with trancode and client userid to determine if encryption is desired.
return_code	Output	A two-character array returning the results of the decryption attempt. The following values are supported: ENCRYPTION_USED—defined as " " ENCRYPTION_SIZE_EXCEEDED_MAX—defined as "01" ENCRYPTION_NOT_USED—defined as "02" EnCRYPTION_APPLICATION_ERROR—defined as "03"
*failureMsg	Output	The pointer to an 80-character array, to be populated by the exit that can receive a null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of ENCRYPTION_APPLICATION_ERROR.

Return Code

None directly, see the preceding return_code structure member.

Default Behavior

The default logic of this user exit is to return ENCRYPTION_NOT_USED.

Building on Windows

The Server Encryption User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

The following are related user exits:

- TIRDCRYP
- WRSECDECRYPT

TIRSECV—Security Validation Exit (Windows)

```
void TIRSECV(char *rp1,
    char *rp2,
    unsigned char Enhanced_Security_Flag,
    PTIRSECV_cmcb pTIRSECV_cmcb);
```

Source Code

TIRSECV.C

Purpose

This security exit is called for every cooperative flow, regardless of the security type used. To facilitate security validation a flag indicating whether the security data is for a standard or enhanced buffer has been added. This exit is intended to provide the opportunity to validate enhanced security data while at the same time not impacting those using standard security. To this effect, the default code provided handles two possible conditions:

- For buffers containing standard security the client userid, client password, and security token fields are expected to be blank. The default behavior is for the exit to return SECURITY_USED, thus indicating that the request is authorized. The exit must be modified to return SECURITY_APPLICATION_ERROR if the intent is that all buffers contain enhanced security data.
- For buffers containing enhanced security the client userid, client password, and security token fields can or cannot contain data. The default behavior is for the exit to return SECURITY_NOT_USED, this indicating that no validation processing was attempted. The exit must be modified to validate the security data and set the relevant return code (return SECURITY_USED for an authorized user and SECURITY_APPLICATION_ERROR for a non authorized user). When returning SECURITY_APPLICATION_ERROR, this exit can provide an optional failure message, using the failure_msgbuffer contained within the TIRSECV_cmcb structure that will be presented to the client.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
Enhanced_Security_Flag	Input	A single character denoting if the CFB has been created to support enhanced security. A value of Y denotes enhanced security,
*pTIRSECV_cmcb	Input	A pointer to a structure containing the following values:
client_userid	Input	A 64-byte character array containing a user ID if the CFB uses enhanced security. For a CFB containing standard security this parameter is expected to be blank.

Name	I/O	Description
client_password	Input	A 64-byte character array containing a password if the CFB uses enhanced security. For a CFB containing standard security this parameter is expected to be blank.
lSecurityTokenLen	Input	A long value representing the length of the pSecurityToken, if any.
pSecurityToken	Input	A pointer to a security token if the CFB uses enhanced security. For a CFB containing standard security this parameter is expected to be blank.
trancode	Input	An 8-byte character array containing the transaction code
return_code	Input/Output	A 2-byte character array containing a value denoting success for failure of this exit. Valid values are: SECURITY_USED - defined as " " SECURITY_NOT_USED—defined as "02" SECURITY_APPLICATION_ERROR—defined as "03"
failure_msg	Input/Output	The pointer to an 80-character array that can be populated by this exit with a null terminated error message string. The string pointed to by this parameter will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of SECURITY_APPLICATION_ERROR.

Return Code

None directly, see the preceding return_code structure member.

Default Behavior

The default logic of this user exit is to return SECURITY_NOT_USED, which is considered an error if this user exit is actually called since the Server Manager requested Client Security validation.

Building on Windows

The Security Validation User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

WRSECTOKEN

TIRXINFO—Locale Information Exit (Windows)

```
void TIRXINFO (char *osId,
    char *codePage,
    long *padChar);
```

Source Code

TIRXLAT.C

Purpose

This exit provides information about the codepage environment of the executing server process. An osId, codepage ID, and default padding character are returned. The runtime uses the osId and codePage returned as parameters passed into the TIRXLAT user exit.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*osId	Output	A pointer to character buffer to contain an OS ID (9 bytes, 8 characters plus NULL terminator). This value will be passed to TIRXLAT as the outOS parameter for inbound transactions and as the inOS parameter for outbound transactions. The current default value is MBCS. This should *not* be confused with an identifier of the underlying operating system on which the server is executing.
*codePage	Output	A pointer to character buffer to contain a codepage ID (9 bytes, 8 characters plus NULL terminator). This value will be passed to TIRXLAT as the outCodePage parameter for inbound transactions and as the inCodePage parameter for outbound transactions. The default value, as returned from this exit, is hard coded into the generated server manager at code generation time. This value will depend upon the platform used to generate the server manager. If the server manager is generated on a Windows platform the value will be 1252, if generated on a UNIX platform using CSE its value will be 819.
*padChar	Output	A pointer to a long value, not currently used for Windows or UNIX servers.

Return Code

None

Default Behavior

The string returned for osId is currently hard coded to a value of MBCS. The value for CodePage is obtained from the server manager. The CodePage number is created during the server manager code generation process. The padChar value is currently unused.

Building on Windows

The Locale Information User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

TIRXLAT

TIRXLAT—National Language Translation Exit (Windows)

```
void TIRXLAT (char *inBuf;
    long *inLen;
    char *inCodePage;
    char *inOS;
    char *outBuf;
    long *outLen;
    char *outCodePage;
    char *outOS;
    long *outPadChar;
    char *workArea;
    long *outCharCnt;
    long *outByteCnt);
```

Source Code

TIRXLAT.C

Purpose

TIRXLAT allows the conversion of textual data based on from/to codepage and operating system information. View data that is passed between the client and server is translated from the client's code page to the server's code page, and vice versa. TIRXLAT uses the client's code page value, which is passed from the client to the server, and the host's code page value to locate a translation table.

This exit is used to translating both the data received from that client and the data to be sent to the client.

When translating data received from the client, the in* parameters correspond to the client data, the out* parameters correspond to the data presented to the server.

When translating data to be sent to the client the in* parameters correspond to the server data to be sent, the out* parameters correspond to the data presented to the client.

If a suitable translation table is not found, the data will be passed back without translation. The user can replace a translation table to customize their environment.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*inBuf	Input	A character pointer to the input buffer to translate
*inLen	Input	A pointer to a long value which is the length inBuf
*inCodePage	Input	A character pointer to the codepage ID of inBuf (8 bytes + 1 NULL).
*inOS	Input	A character pointer to the OS ID of inBuf (8 bytes + 1 NULL).
*outBuf	Input/Output	A character pointer to the buffer in which to place the translated text
*outLen	Input	A pointer to a long value that is the length of the data pointed to by outBuf.
*outCodePage	Input	A character pointer to the codepage ID corresponding to the output buffer, outBuf.
*outOS	Input	A character pointer to the OS ID corresponding to the output buffer, outBuf.

Name	I/O	Description
*outPadChar	Input	A pointer to a long value which is the padding character to use, 0 if no padding to be done in the output buffer, outBuf.
*workArea	Input	A character pointer to a 100-byte scratch work area.
*outCharCnt	Input/Output	A pointer to a long value which is the number of characters placed into the output buffer, outBuf.
*outByteCnt	Input/Output	A pointer to a long value which is the number of bytes placed into the output buffer, outBuf.

Return Code

None

Default Behavior

If a suitable translation table is found, the data will be translated from the inCodePage to the outCodePage. If a suitable translate table is not found the data is passed back without translation.

Building on Windows

The National Translation User Exit is built as part of the dynamic link library AEUEXITxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the batch file MKEXITS.BAT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Run MKEXITS.BAT.

Related User Exits

TIRXINFO

Windows Server Middleware User Exits

WebSphere MQ Server Transport - Windows User Exits

All supplied WebSphere MQ Server Transport user exits are written using the C programming language. The following table briefly describes the WebSphere MQ Exits:

WebSphere MQ Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_MQS_DPC_SETUPCOMM_COMPLETE	cimqsvex.c	Verifies that connection to target server is successful.
CI_MQS_DPS_EXIT	cimqsvex.c	MQ Directory Services Exit
CI_MQS_DYNAMICQNAME_EXIT	cimqsvex.c	Provide Queue Name that will be used when opening a dynamic queue.

Details for the preceding user exits follow in a separate section for each.

CI_MQS_DPC_SETREPORTOPTIONS – Override Put Queue Report Options Exit (Windows)

```
void CI_MQS_DPC_setReportOptions(MQLONG *reportOptions);
```

Source Code

CIMQSVEX.C

Purpose

This exit can be used to override the set of report options defined for the WebSphere MQ Put Message Descriptor prior to the issuance of an MQPUT() operations.

The report options set by the runtime are described in the following table:

Report Option	**Description**
MQRO_EXCEPTION	This type of report can be generated when an exception occurs. For instance if a message is sent to another queue manager and the message cannot be delivered to the specified destination queue.

Report Option	Description
MQRO_EXPIRATION	An Expiration report. This type of report is generated by the queue manager if the message is discarded prior to delivery to an application because its expiry time has passed.
MQRO_PASS_MSG_ID	If a report is generated, the MsgId of the current message being processed is to be copied to the MsgId of the report message.
MQRO_COPY_MSG_ID_TO_CORREL_ID	Indicates the correlation ID of the report generated should equal the message ID of the request originally issued.
MQRO_DEAD_LETTER_Q	This option causes the original message to be placed on the dead-letter queue when an exception occurs.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*reportOptions	Input/Output	A pointer to a long value representing the currently defined report options to be used in the Put Message Descriptor.

Return Code

None

Default Behavior

The runtime specified report options are left unchanged.

Building on Windows

The WebSphere MQ Override Put Queue Report Options exit is built as part of the dynamic link library MQSSXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile SMQSEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F SMQSEXIT.NT.

Related User Exits

None

CI_MQS_DPS_EXIT—MQSeries DPS Directory Services Exit (Windows)

```
void CI_MQS_DPS_Exit (char   *qMgr,
    char   *srvQname,
    long   *numRequests,
    long   *getMsgTO,
    char   *serverName);
```

Source Code

CIMQSVEX.C

Purpose

The provided sample WebSphere MQ DPS Directory Services exit is an implementation of Transaction routing. Transaction routing is a conceptual process that lets cooperative flow data be routed from a Distributed Process Server (DPS) to a programmatically determined Distributed Process Server (DPS). This exit can use the input serverName, which is the server load module name calling this exit, to programmatically modify the output parameters.

The current cooperative request's local queue manager and put queue names can be overridden using this exit. Additionally a Get queue timeout value as well as a parameter specifying the servers multiple transaction behavior can be customized.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*qMgr	Input/Output	A character string containing the name of the local queue manager. By default, the application obtains this information from the model during generation. This exit can override this name.
*srvQName	Input/Output	A character string containing the name of the put queue connecting to the local queue manager. By default, the application obtains this information from the model during generation. This exit can override this name.
*numRequests	Input/Output	Specifies the number of transactions the server calling this exit can execute prior to the server shutting down. A mechanism to limit the number of transactions a server can execute. Default: -1, no limit to the number of transactions that can be serviced.
*getMsgTO	Input/Output	A long value representing the reply timeout applied to the get queue associated with the current flow request. Default: The default value is MQWI_UNLIMITED (-1), wait indefinitely. If set, the number represents milliseconds. This exit can override this name.
*serverName	Input	A character string containing the name of the server load module executing this exit.

Return Code

None

Default Behavior

The supplied sample does not implement dynamic transaction routing. For more information about default values for the various parameters, see the preceding Arguments section.

Building on Windows

The WebSphere MQ DPS Directory Services exit is built as part of the dynamic link library MQSSXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile SMQSEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F SMQSEXIT.NT.

Related User Exits

None

CI_MQS_DYNAMICQNAME_EXIT—Dynamic Queue Name Override Exit (Windows)

```
void CI_MQS_DynamicQName_Exit (char   *dynamicQName);
```

Source Code

CIMQSVEX.C

Purpose

The Dynamic Queue Name exit allows override of the queue name that will be used when opening a dynamic queue. The resulting Dynamic Queue will obtain its attributes from the specified WebSphere MQ Model Queue name. The passed in dynamicQName area can be modified by placing a null terminated string of the value to be used as the dynamic queue name, including the use of valid WebSphere MQ pattern characters used to name dynamic queues.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*dynamicQName	Input/Output	A pointer to a character buffer, of length MQ_Q_NAME_LENGTH+1 (48 +1), that contains the default name of the dynamic queue as built by the WebSphere MQ runtime (that is, username.processid.threadid).

Return Code

None

Default Behavior

The dynamic queue name is not modified.

Building on Windows

The WebSphere MQ DPS Dynamic Queue Name Override exit is built as part of the dynamic link library MQSSXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile SMQSEXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F SMQSEXIT.NT.

Related User Exits

None

Windows C Proxy User Exits

The following table summarizes the functions available through the user exits for C Proxy applications:

C Proxy: Language: C		
User Exit Name	**Source Code**	**Description**
WRSECTOKEN	proxyxit.c	Client Security Token Exit
WRSECENCRYPT	proxyxit.c	Client/Server Encryption Exit
WRSECDECRYPT	proxyxit.c	Client/Server Decryption Exit

C Proxy user exits are rebuilt into the DLL PREXxxN.DLL using the command procedure proxyxit.nt in %GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit.

Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

WRSECTOKEN—Client Security Token Exit (Windows)

```
int WRSECTOKEN (char *clientUserid,
    char *clientPassword,
    char *trancode,
    char *nextLocation,
    BOOL *bClntMgrSecurity,
    long *tokenLen,
    char *token,
    char *failureMsg)
```

Source Code

PROXYXIT.C

Purpose

The Client Side Security Exit is invoked by the proxy runtime to let a user influence how client security data is processed by the proxy runtime code involved in servicing a cooperative flow. Specifically, this exit influences if the Common Format Buffer (CFB) request will contain a security offset and if that data populated in the security offset should be used by other runtime components such as the Client Manager or Communications Bridge when servicing the cooperative flow request.

The trancode and nextLocation variables are provided as input. These input values can be used by the user exit code to determine what return code value should be specified.

In addition to the return code value, this exit has the option of returning some fields as output data to the calling runtime code.

For more information about the input and output fields of this exit routine, see Arguments. For a description on what the invoking proxy runtime will do because of receiving one of the expected return values, see Return Codes.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*clientUserid	Input/Output	A pointer to a character array that contains the value of the proxy ClientUserid attribute. This user exit can set this value by modifying the data area pointed to by this argument. The value assigned by this user exit cannot exceed 64 bytes.
*clientPassword	Input/Output	A pointer to a character array that contains the value of the proxy ClientPassword attribute. This user exit can set this value by modifying the data area pointed to by this argument. The value assigned by this user exit cannot exceed 64 bytes.
*trancode	Input	A pointer to a character array that contains the trancode associated with the flow being processed by the proxy runtime synchronous or asynchronous cooperative flow operation.
*nextLocation	Input	A pointer to a character array that contains the Next Location variable associated with the flow being processed by the proxy runtime synchronous or asynchronous cooperative flow operation.
*bClntMgrSecurity	Output	A pointer to an integer Boolean field that can be set to either TRUE or FALSE. The value of this field only has meaning if this user exit returns SecurityUsedEnhanced. TRUE indicates that the security data (Client User ID and Client Password) that is added to the security offset of the associated CFB should be used as the source of the UserID and Password by the Client Manager or Communications Bridge.

Name	I/O	Description
*tokenLen	Input/Output	On input, tokenLen is a pointer to a long integer field that contains the maximum length of the allocated token character buffer. The maximum token length is dependent on the available space remaining during the construction of the CFB. On return from the exit, the long integer pointed to by tokenLen should contain the actual length of data returned in the character array, which is pointed to by the token argument. **Note:** The use of a token is optional, and therefore, setting the long integer pointed to by tokenLen to zero indicates that a token is not specified by the user exit. The length value returned by this field only has meaning if this user exit returns SecurityUsedEnhanced.
*token	Input/Output	On input, token is a pointer to a character array that will accept a user specified security token. The use of a user specified security token is optional. The token data that is provided by this user exit will be provided to the server side TIRSECV security user exit. The security token returned by this field only has meaning if this user exit returns SecurityUsedEnhanced.
*failureMsg	Input/Output	The pointer to an 80-character array that can receive a user provided null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is displayed by the proxy runtime.

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
SecurityNotUsed	Indicates to the runtime that the CLIENT_USER_ID, CLIENT_PASSWORD, and security token will NOT be used to populate any part of the cooperative flow request. The client side security variables will not be added to the CFB by the proxy runtime.

Return Code	Description
SecurityUsedStandard	Indicates to the runtime that at most eight (8) bytes of the CLIENT_USER_ID and at most eight (8) bytes of the CLIENT_PASSWORD data will be set into the CFB header. The associated request buffer will not contain a CFB Security Offset area, and will therefore, not contain a security token. Additionally, by not making use of the CFB Security Offset area, the Client User ID and Client Password values are not eligible for being encrypted.
SecurityUsedEnhanced	Indicates to the runtime that the CLIENT_USER_ID, CLIENT_PASSWORD, and the optional Security Token should be added to the CFB by way of the CFB Security Offset. Additionally, at most (8) bytes of the Client User ID value will be set into the CFB header.
SecurityError	Indicates to the runtime that an error was encountered by the user exit and that the processing of the associated request has failed. The error indication and message string returned using the failureMsg argument would be returned to the proxy runtime. The proxy runtime will popup an error message display indicating the failed request.

Default Behavior

The WRSECTOKEN user exit, as delivered with CA Gen, will return SecurityNotUsed. In addition, although not necessary, the user exit will set the long integer pointed to by the tokenLen pointer to zero, and set the Boolean field pointed to by the bClntMgrSecurity pointer to False.

Building on Windows

The C Proxy Decryption exit is built as part of the dynamic link library PREX*xx*N.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile PROXYXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F PROXYXIT.NT.

Related User Exits

The following are related user exits:

- TIRSECV
- WRSECENCRYPT
- WRSECDECRYPT

WRSECENCRYPT—Client Side Encryption Exit (Windows)

```
int WRSECENCRYPT (char *trancode,
    char *nextLocation,
    char *clientUserid,
    long maxViewLen,
    long *encryptViewLen,
    unsigned char *encryptView,
    char *failureMsg)
```

Source Code

PROXYXIT.C

Purpose

The Client Side Encryption exit is called by the proxy runtime to provide the opportunity to encrypt a cooperative flow request from C Proxy applications. The data in the Common Format Buffer (CFB) that is eligible to be encrypted include the cooperative flow's view data and optional security offset area.

The user provides an encryption algorithm that consists of manipulating the data pointed to by encryptView. The encryptViewLen, on input contains the number of bytes eligible for being encrypted. The process of encryption cannot result in an encrypted buffer area that exceeds maxViewLen. If encryption is performed by this exit, EncryptViewLen must be updated with the length of the encrypted result. Additionally, this exit must return the EncryptionUsed return code value.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*trancode	Input	A pointer to a character array that contains the trancode associated with the synchronous or asynchronous cooperative flow being processed by the proxy runtime.
*nextLocation	Input	A pointer to a character array that contains the Next Location associated with the synchronous or asynchronous cooperative flow being processed by the proxy runtime.
*clientUserid	Input	A pointer to a character array that contains the value of the of the CLIENT_USER_ID variable associated with the flow being processed by the proxy runtime synchronous or asynchronous cooperative flow processing. The CLIENT_USESRID variable is optionally set by Action Language coded within the C Proxy code.
MaxViewLen	Input	A long field that contains the maximum available buffer space (in bytes) that the encrypted data can occupy.
*encryptViewLen	Input/Output	A pointer to a long field. On input, EncryptViewLen is the length of the current buffer space (in bytes) of the data eligible for being encrypted. On output, EncryptViewLen should be updated to contain the length of the encrypted data. The length of the encrypted result cannot exceed maxViewLen.
*encryptView	Input/Output	A character pointer to the starting location of the data eligible for being encrypted. The encrypted data must be copied to this same memory location.

Name	I/O	Description
*failureMsg	Input/Output	The pointer to an 80-character array that can receive a user provided null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is displayed by the proxy runtime.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
EncryptionNotUsed	Indicates to the runtime that the user exit did not perform any encryption to the provided data buffer.
EncryptionUsed	Indicates to the runtime that the user exit did perform encryption on the provided data. The runtime marks the CFB as being encrypted. An encrypted CFB will trigger the decryption counterpart user exit to be invoked by the target server manager. The server side decryption user exit is TIRDCRYP.
EncryptionFailure	Indicates to the runtime that an error was encountered by the user exit and that the processing of the associated request has failed. The error indication and message string returned using the failureMsg argument would be returned to the proxy runtime. The proxy runtime will pop up an error message display indicating the failed request.

Default Behavior

The WRSECENCRYPT user exit, as delivered with CA Gen, will return EncryptionNotUsed.

Building on Windows

The C Proxy Security Token exit is built as part of the dynamic link library PREX*xx*N.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile PROXYXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F PROXYXIT.NT.

Related User Exits

The following are related user exits:

- TIRNCRYP
- WRSECDECRYPT

WRSECDECRYPT—Client Decryption Exit (Windows)

```
int WRSECDECRYPT (long maxViewLen,
    long *encryptViewLen,
    unsigned char *encryptView,
    char *failureMsg)
```

Source Code

PROXYXIT.C

Purpose

The Client Side Decryption exit is called by the proxy runtime when an encrypted response buffer is received from a target server.

The user provides a decryption algorithm that manipulates the data pointed to by encryptView. The encryptViewLen, on input contains the number of bytes available into which the encrypted buffer area can be decrypted. The process of decryption cannot result in a decrypted buffer area that exceeds maxViewLen. If decryption is performed by this exit, EncryptViewLen must be updated with the length of the decrypted result.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
maxViewLen	Input	A long field that contains the maximum available buffer space (in bytes) that the decrypted data can occupy.
*encryptViewLen	Input/Output	On input, EncryptViewLen is the current buffer space (in bytes) of the encrypted data. On output, EncryptViewLen should be updated to contain the length of the decrypted data. The length of the decrypted result cannot exceed maxViewLen.
*encryptView	Input/Output	A pointer to the starting location of the data eligible for being decrypted. The decrypted data must be copied back into this same memory location.
*failureMsg	Input/Output	The pointer to an 80-character array that can receive a user provided null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is displayed by the proxy runtime.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
DecryptionNotUsed	Indicates to the runtime that the user exit did not perform any decryption of the encrypted data buffer.
DecryptionUsed	Indicates to the runtime that the user exit successfully performed the decryption of the provided encrypted data.

Return Code	Description
DecryptionFailure	Indicates to the runtime that an error was encountered by the user exit and that the decryption processing has failed The error indication and message string returned using the failureMsg argument will be returned to the proxy Runtime. The proxy runtime will pop up an error message display indicating the failed request.

Default Behavior

The WRSECDECRYPT user exit, as delivered with CA Gen, will return DecryptionNotUsed.

Building on Windows

The C Proxy Encryption exit is built as part of the dynamic link library PREX*xx*N.DLL.

Note:*xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

A prerequisite for building the DLL, you must have Microsoft's Visual C++ compiler installed on your system.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile PROXYXIT.NT (%GEN*xx*%Gen\VSabc for Visual Studio 32-bit or %GEN*xx*%Gen\VSabc\amd64 for Visual Studio 64-bit).

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F PROXYXIT.NT.

Related User Exits

The following are related user exits:

- TIRDCRYP
- WRSECENCRYPT

Chapter 3: UNIX and Linux User Exits

There are several sets of user exits to support the variety of C applications that run on the UNIX and Linux platforms. Scripts are provided to assist in building the runtime user exits for each runtime environment listed in the following sections.

The following table lists the sets of C user exits available on the UNIX and Linux platforms:

User Exit Set	Provided As
Blockmode Runtime	libae_userexits_c.*, libae_db2.*, libae_ora.*
Client Middleware	
Common System Utilities (CSU)	libcsuvn.xx.*
TCP/IP as Transport	libtcpcx.xx.*
WebSphere MQ as Transport	libmqscx.xx.*
Tuxedo as Transport	libtxwcx.xx.*, libtxcx.xx.*
Web Services as Transport	libwscx.xx.*
Server Runtime	libae_userexits_c.*, libae_db2.*, libae_ora.*
Asynchronous Daemon	aefsecex
Server Middleware	
WebSphere MQ as Transport	libmqssx.xx.*
Tuxedo as Transport	libtxsx.xx.*
C Proxy	libprex.xx.*

Note: * refers to the shared library extension. Each UNIX and LINUX system has different file extensions for shared libraries. Refer to your CA Gen installation for the shared library extension used.

UNIX and Linux Blockmode User Exits

The following table summarizes the functions available through the user exits for generated blockmode applications:

Name	Description
DBCOMMIT	Database Commit User Exit. There is one user exit routine for each supported database: Oracle, and DB2.
DBCONNCT	Database Connection User Exit. There is one user exit routine for each supported database: Oracle, and DB2.
DBDISCNT	Database Disconnect User Exit. There is one user exit routine for each supported database: Oracle, and DB2.
TIRDLCT	Dialect User Exit
TIRDRTL	Default Retry Limit User Exit
TIRHELP	Help Interface User Exit
TIRMTQB	Message Table User Exit
TIRSECR	Security Interface User Exit
TIRSYSID	System ID User Exit
TIRTERMA	User Termination User Exit
TIRTIAR	Database error message User Exit. There is one user exit routine for each supported database: Oracle and DB2.
TIRUPDB	MBCS Uppercase Translation User Exit
TIRUPPR	Uppercase Translation User Exit
TIRURTL	Ultimate Retry Limit User Exit
TIRUSRID	User ID User Exit
TIRYYX	Date User Exit

Note: The database user exits DBCONNCT, DBCOMMIT, DBDISCNT, and TIRTIAR are rebuilt into individual shared libraries (libae_db2.*, libae_ora.*) using the script $IEFH/make/mkdbs.

Blockmode runtime user exits are rebuilt into the shared library libae_userexits_c.* using the script $IEFH/make/mkexits. This is the same shared library that is used with Server applications.

Details for the preceding user exits follow in a separate section for each.

DBCOMMIT—Database Commit Exit

```
void dbcommit (

int rc)
```

Source Code

For Db2: tirdb2.ppc

For Oracle: tirora.ppc

Purpose

Use the Database Commit User Exit to customize the commit logic for a particular database. The default processing of this user exit provides a simple database commit.

There exists a Database Commit User Exit for each supported DBMS: Oracle and DB2.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
rc	Input	Type of commit to perform

Return Code

None

Default Behavior

By default, these modules perform a standard database commit statement.

Building on UNIX/Linux

The Database Commit User Exit is built as part of the shared library libae_db2.* and/or libae_ora.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).

3. Run mkdbs.
4. You will be prompted for the DBMS choice and a version choice if the DBMS is Oracle (the currently supported version is the default).

   The DBMS user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

DBCONNCT, DBDISCNT, TIRTIAR

DBCONNCT—Database Connection User Exit

```
int dbconnect (
    char *user,
    char *pswd)
```

Source Code

For Db2: tirdconn.sqc

For Oracle: tiroconn.sqc

Purpose

Use the Database Connection User Exit to customize the connection to the particular database. This user exit enhances database security. The default processing of this user exit provides a simple database connection.

There exists a Database Connect User Exit for each supported DBMS: Oracle and DB2.

The default method for CA Gen to acquire DBMS connection information for blockmode applications is for the AEF to locate the trancode in the AEENV file. Connection information includes the username, password, and database name. Each person that executes the application must have read access to the AEENV file that contains the connection information.

Database	Description	Host Variable	Declared Type
Oracle	user ID	uid	VARCHAR(32)
Oracle	Password	pwd	VARCHAR(32)
DB2	user ID	uid	char(9)
DB2	Password	pwd	char(9)
DB2	database name	dbname	char(9)

We recommend leaving the call to dbid() unchanged, and adding logic immediately before the database connect statements to populate the appropriate variables. Ensure that you add all code that the DBMS requires. For example, verify arr and len elements are populated correctly for VARCHAR. We also recommend that all AEENV files contain character strings as place holders for the database connection information. These character strings do not have to contain valid connection information.

For greater security, add a call to an encryption routine.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*user	Input	Pointer to DBMS userid
*pswd	Input	Pointer to DBMS password for userid

Return Code

Integer representing success or failure of database connection.

Default Behavior

By default, these modules read the database connect information from the AEENV file and use the information in the database connect statement.

Building on UNIX/Linux

The Database Connection User Exit is built as part of the shared library libae_db2.* and/or libae_ora.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkdbs.
4. You will be prompted for the DBMS choice and a version choice if the DBMS is Oracle (the currently supported version is the default).

The DBMS user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

DBCOMMIT, DBDISCNT

DBDISCNT—Database Disconnect Exit

```
int dbdiscnt()
```

Source Code

For Db2: tirdb2.ppc

For Oracle: tirora.ppc

Purpose

Use the Database Disconnect User Exits to customize the database disconnect. The default processing of these users exits provides simple database disconnect.

There exists a Database Disconnect User Exit for each supported DBMS: Oracle and DB2.

Arguments

None

Return Code

Integer representing success or failure of database disconnection.

Default Behavior

By default, these modules perform a standard database disconnect statement.

Building on UNIX/Linux

These user exits are built as part of the shared library libae_db2.* and/or libae_ora.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).

3. Run mkdbs.
4. You will be prompted for the DBMS choice and a version choice if the DBMS is Oracle (the currently supported version is the default).

The DBMS user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

DBCONNCT, DBCOMMIT

TIRDLCT—Dialect Exit

```
void TIRDLCT (
    char *rp1,
    char *rp2,
    struct dialect_cmcb *tirdlct_cmcb)
```

Source Code

TIRDLCT

Purpose

TIRDLCT supplies the current user's dialect to the application and is useful only for multilingual applications. For multilingual support, the user is responsible for modifying this module to return the appropriate dialect. The dialect returned should be defined using the Design selection on the CA Gen action bar. If it is not, the application's default dialect is used.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*tirdlct_cmcb	Input/Output	A pointer to a structure containing the following items:
tirdlct_userid	Input	An 8-byte character array containing the current user id as provided by TIRUSRID.

Name	I/O	Description
tirdlct_terminal_id	Input	An 8-byte character array containing the current terminal id.
tirdlct_system_id	Input	An 8-byte character array containing the current system id as provided by TIRSYSID.
tirdlct_return_dialect	Input	An 8-byte character array containing the returned dialect.

Return Code

None

Default Behavior

TIRDLCT returns a dialect value of DEFAULT.

Building on UNIX/Linux

The Dialect User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

TIRUSRID, TIRSYSID

TIRDRTL—Default Retry Limit Exits

```
int tirdrtl (
    char retry_flag)
```

Source Code

TIRDRTL.C

Purpose

TIRDRTL lets you override the CA Gen-defined default value for the TRANSACTION RETRY LIMIT system attribute. TRANSACTION RETRY LIMIT will be initialized to this value at the beginning of each new transaction. This value can subsequently be modified by a SET TRANSACTION RETRY LIMIT statement in an action diagram.

TRANSACTION RETRY LIMIT is used to specify the maximum number of times to retry a transaction when one of the following events occurs:

- A RETRY TRANSACTION action diagram statement executes.
- A deadlock or timeout occurs trying to access a database, and there is no WHEN DATABASE DEADLOCK OR TIMEOUT statement for that entity action statement.

In these cases, uncommitted database updates are rolled back, and an attempt is made to execute the application again. After the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit User Exit (see TIRURTL), no more retries can occur, and the application fails with a runtime error.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
retry_flag	Input	Flag to indicate whether or not to set a retry limit.

Return Code

Integer containing the retry limit.

Default Behavior

If the Default Retry Limit User Exit is not used, TRANSACTION RETRY LIMIT will be initialized to 10 for all target environments. If the Default Retry Limit User Exit is used, it must not return a value greater than that specified in the Ultimate Retry Limit User Exit.

Building on UNIX/Linux

The Default Retry User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

TIRURTL

TIRHELP—Help Interface Exit

```
void TIRHELP (
    char *rp1,
    char *rp2,
    struct tirhelp *in_tirhelp_cmcb,
    char *in_tirhelp_return_message,
    char *in_environment_list,
    char *in_application_list,
    struct scmgr *in_scmgr_cmcb)
```

Source Code

TIRHELP.C

Purpose

TIRHELP is called when a HELP or PROMPT command is entered. From TIRHELP, a help system can be invoked to provide application help information.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*in_tirhelp_cmcb	Input/Output	A pointer to a structure containing the following items:
tirhelp_request_code	Input/Output	A 2-byte character array containing the type of help requested.
tirhelp_return_code	Output	A 2-byte character array containing the return code.
tirhelp_mapname	Input	An 8-byte character array containing the mapname.
tirhelp_data_filler	Unused	An 8-byte character array used as a structure filler.
tirhelp_trancode	Input	An 8-byte character array containing the trancode.
tirhelp_userid	Input	An 8-byte character array containing the user id.
tirhelp_terminal_id	Input	An 8-byte character array containing the terminal id.
tirhelp_printer_id	Input	An 8-byte character array containing the printer id.
tirhelp_dialect	Input	An 8-byte character array containing the dialect.
tirhelp_message_table	Input	An 8-byte character array containing the message table. This value is passed to TIRMTQB.
tirhelp_filler	Unused	A 16-byte character array used as a structure filler.
tirhelp_last_command	Input	An 80-byte character array containing the last command.
tirhelp_last_message	Input	An 80-byte character array containing the last message.
tirhelp_screen_helpid	Output	A 44-byte character array containing the help identifier for the screen.
tirhelp_field_helpid	Output	A 44-byte character array containing the help identifier for the field.
tirhelp_field_token1	Input	A 3-byte character array containing a field token.

Name	I/O	Description
tirhelp_field_token2	Input	A 3-byte character array containing a second field token
tirhelp_field_len	Input	A 3-byte character array containing the field length.
tirhelp_field_value	Input	A 256-byte character array containing the value of the field.
tirhelp_field_protect	Input	A single character containing a field protection flag.
tirhelp_field_intens	Input	A single character containing a field intensity flag.
in_tirhelp_return_message	Output	An 80-byte character array representing the returned help message. By default, this message is returned from a call to TIRMTQB.
in_environment_list	Input	A pointer to an environment control block. Reserved for runtime internal use only.
in_application_list	Input	A pointer to an application control block. Reserved for runtime internal use only.
in_scmgr_cmcb	Input/Output	A pointer to a screen management control block.

Return Code

None

Default Behavior

The TIRHELP routine will return a message indicating no help is available.

Building on UNIX/Linux

The Help Interface User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

TIRMTQB

TIRMTQB—Message Table Exit

```
void TIRMTQB(char *rp1,
    char *rp2,
    char *msg_tbl_name,
    short *msgnum,
    struct PARMMSG *prm);
```

Source Code

TIRMTQB.C

Purpose

This message table exit is called by the runtime when a system-level message is to be displayed. The user can customize the wording of the messages within this exit. Additional tables can also be defined to support other dialects.

The default table includes an entry for each CA Gen runtime error message. Each entry includes the following information:

- Message Number—The message number is permanently assigned by CA Gen. Each message has a unique number.
- Message Text—The message text is the actual words that appear on the application screen when an error occurs. The message text, and any variable values that can be appended, is truncated if it exceeds the length of the error message line defined for the application screen. The error message line is a maximum of 80 characters of which 12 are reserved for the message number.

If the message number is not in the table, TIRMTQB returns a default message.

Runtime Error Table

The Runtime Error Message Table includes an entry for each runtime error message. Each table entry includes the following information:

- Message type— a message number is not found in the table, or when you return to a transaction screen after a fatal error or a Dialog Manager error occurs. Valid message types are shown in the following list:
- Default message— a message number is not found in the table, or when you return to a transaction screen after a fatal error or a Dialog Manager error occurs.

- Dialog Manager error—Occurs when the Dialog Manager is unable to communicate with the system. This is a fatal error beyond the control of CA Gen. An error in the load module packaging or in the configuration specifications causes a Dialog Manager error. Error handling is the same as for a fatal error.
- Fatal error— a CA Gen application abnormal program ending. If a condition occurs at runtime that the generated code cannot handle, the system issues a fatal error. An error message screen displays the appropriate error messages.
- Function error—Occurs if a CA Gen-supported function receives invalid input or produces invalid output. CA Gen-supplied functions manipulate characters, numbers, dates, and times.
- Screen edit error—A non-fatal error that occurs when an input or output value for a field does not match the expected value, the range, type, or format defined for the field during model development. This type of message is displayed on your transaction screen. You can correct the error and continue with the transaction.
- Unformatted input error—Occurs when the unformatted input contains invalid parameters, delimiters, or both. Unformatted input is a list of parameters associated with a clear screen transaction code.
- Message number—Each message has a unique number that is permanently assigned by CA Gen.
- Message text—The message text consists of the actual words that appear on the application screen when an error occurs. Because of the length of the message identifier, the message text is limited to 68 characters for an 80-character screen. The message text and appended variables are truncated if they exceed the length of the error message line defined for the application screen.
- Suffix—(If applicable) The suffix contains variable values, such as return codes, permitted values, or the values in error.

Runtime Error Handling

Runtime errors are handled by the Dialog Manager. Runtime errors are non-fatal, such as screen edit, or fatal errors.

If a non-fatal error such as invalid user input occurs, the Dialog Manager displays an error message on the transaction screen. You can correct the error and continue processing the transaction.

If an application fails because of a fatal error, transaction processing terminates, and the error processing is as follows:

- The Dialog Manager performs all necessary rollbacks of the databases.
- CA Gen displays an error message screen that lists the appropriate runtime error messages.
- Pressing Enter from the error message screen causes CA Gen applications to terminate execution.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*msg_tbl_name	Input	A character string containing the name of the table to be used for extraction of the message text. Currently one table named DEFAULT is supported by the CA Gen runtime.
*msgnum	Input	A short value containing the message number corresponding to the text to be fetched.
*prm	Input/ Output	A pointer to a PARMMSG structure to contain the returned message text information. This structure, defined in tirmtq.h, has the following definition:
PARMLEN		A short value containing the total length of PARMNO + PARMTXT.
PARMNO	Output	An 11-byte character array containing the message number formatted in a standard style.
PARMTXT	Output	A string containing the text corresponding to the error message number. The string can be up to 245 bytes, including the terminating NULL.
filler	Unused	Two unused filler characters

Return Code

None

Default Behavior

The table in the default exit is used to retrieve runtime error message text.

Building on UNIX/Linux

The Message Table User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

TIRSECR—Security Check Interface Exit

```
void TIRSECR(char * rp1,
    char * rp2,
    struct security_cmcb * in_tirsecr_cmcb);
```

Source Code

TIRSECR.C

Purpose

The Dialog Manager calls the Security Check Interface Exit when a transaction is started and before execution of a dialog flow. This allows transaction-level security checking to be implemented. The following data is provided by the dialog manager of each load module for use in checking security authorization:

- System ID (as provided by the System ID Exit, TIRSYSID)
- User ID (as provided by the User ID Exit, TIRUSRID)
- Trancode
- Terminal ID

- Load module name
- Procedure step name

If the user defined security check passes, TIRSECR should move a value of spaces to the return code. If the security check fails, a non-blank value should be moved to the return code with a message describing the violation inserted into the tirsecr_failure_msg buffer. The current dialect in effect on the client is passed in using tirsecr_dialect.

When the dialog manager receives control, it proceeds with the transaction if the return code is spaces, or issues an error if it is not.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*in_tirsecr_cmcb	Input/Output	A pointer to a structure containing the following items:
tirsecr_userid	Input	An 8-byte character array containing the security user ID as provided by the TIRUSRID user exit
tirsecr_trancode	Input	An 8-byte character array containing the current transaction code.
tirsecr_terminal_id	Input	An 8-byte character array containing the current terminal ID.
tirsecr_system_id	Input	An 8-byte character array containing the current system ID as returned by the TIRSYSID user exit.
tirsecr_load_module	Input	An 8-byte character array containing the name of the executing load module calling this exit.
tirsecr_pstep_name	Input	A 32-byte character array containing the name of procedure step being executed.
tirsecr_dialect	Input	A 32-byte character array containing the dialect in effect on the client.
tirsecr_return_code	Output	A 2-character array representing the success or failure of this exit processing. TIRSECR_ALL_OK defined as two spaces implies success, any other value implies failure. If none spaces are return, tirfail will be passed the tirsecr_failure_msg message.

Name	I/O	Description
tirsecr_failure_msg	Output	An 80-byte character array used in conjunction with a failing return code in tirsecr_return_code. This exit can insert an error message into this array that will be passed by the Dialog manager to the tirfail user exit.

Return Code

None directly. For more information, see tirsecr_return_code structure member.

Default Behavior

The default exit will return a status code of spaces, indicating no security violation was detected.

Building on UNIX/Linux

The Security Check User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

TIRUSRID, TIRSYSID

TIRSYSID—System ID Exit

```
void TIRSYSID ( char *rp1;
    char *rp2;
    char *system_id);
```

Source Code

TIRSYSID.C

Purpose

TIRSYSID supplies the system ID to the application.

The purpose of TIRSYSID is to implement application logic that lets you implement one model on multiple platforms, and perform processing appropriate for the platform. The system ID is also one of the parameters passed to the Security Interface Exit (TIRSECR).

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*system_id	Output	An 8-byte character array representing the system identifier where the server application is executing.

Return Code

None

Default Behavior

By default, TIRSYSID calls the runtime routine DEFSYSID. This routine returns a default system ID, the value of which depends on the platform on which the application is executing.

Under UNIX/Linux if the environment variable IEF_SYSID is set the first 8 characters of this variable are used. Otherwise, “UNIX“ is returned.

Building on UNIX/Linux

The System ID User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

TIRSECR

TIRTERMA—User Termination Exit

```
void TERTERMA (
    char *rp1,
    char *rp2,
    struct term_pb *pb)
```

Source Code

TIRTERMA.C

Purpose

TIRTERMA is called when an application fails. Modification of TIRTERMA lets the user customize the handling of runtime errors.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.

Name	I/O	Description
*pb	Input/ Output	A pointer to a PARMMSG structure to contain the termination information. This structure is defined in tirterma.h.

Return Code

None

Default Behavior

The default processing for TIRTERMA returns a status code of spaces, indicating to use standard error handling.

Building on UNIX/Linux

The User Termination User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

TIRTIAR—Database Error Message Exit

```
void TIRTIAR (
    char *sqlca,
    short *ml,
    char *mb,
    int maxLength)
```

Source Code

For Db2: tirdb2.ppc

For Oracle: tirora.ppc

Purpose

Use the Database Error Message User Exit to customize the error message received from the database commit. The default processing of this user exit provides a simple database error message.

There exists a Database Error Message User Exit for each supported DBMS: Oracle and DB2.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*sqlca	Input	Pointer to SQLCA structure
*ml	Output	Pointer to error message length
*mb	Output	Pointer to error message buffer
maxLength	Input	Maximum length of error message that can be written to error message buffer

Return Code

None

Default Behavior

By default, these modules provide the error message returned by the database commit.

Building on UNIX/Linux

The Database Error Message User Exit is built as part of the shared library libae_db2.* and/or libae_ora.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkdbs.
4. You will be prompted for the DBMS choice and a version choice if the DBMS is Oracle (the currently supported version is the default).

The DBMS user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

DBCOMMIT

TIRUPDB—MBCS Uppercase Translation Exit

```
void TIRUPDB (
    char *rp1,
    char *rp2,
    char *tbl_name,
    long *len,
    char *xlate_data)
```

Source Code

TIRUPDB.C

Purpose

TIRUPDB is called to uppercase multi-byte text. The user can modify the mechanism used to uppercase multi-byte text with this user exit.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*tbl_name	Input	A pointer to a translation table name.
*len	Input/Output	Length of text to convert to uppercase.
*xlate_data	Input/Output	A pointer to the text to be uppercased.

Return Code

None

Default Behavior

The default translation uses MBCS functions to perform uppercase translation based upon the active system code page. However, the system designer, programmer, may add code to recognize dialects and perform any lower to upper functionality desired. In that case, insure that the default behavior still uses the MBCS libraries.

Building on UNIX/Linux

The MBCS Uppercase Translation User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

TIRUPPR

TIRUPPR—Uppercase Translation Exit

```
void TIRUPPR (
    char *rp1,
    char *rp2,
    char *tbl_name,
    long *len,
    char *xlate_data)
```

Source Code

TIRUPPR.C

Purpose

TIRUPPR is called to uppercase multi-byte text. The user can modify the mechanism used to uppercase multi-byte text with this user exit.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*tbl_name	Input	A pointer to a translation table name.
*len	Input/Output	Length of text to convert to uppercase.
*xlate_data	Input/Output	A pointer to the text to be uppercased.

Return Code

None

Default Behavior

The default translation uses MBCS functions to perform uppercase translation based upon the active system code page. However, the system designer, programmer, may add code to recognize dialects and perform any lower to upper functionality desired. In that case, insure that the default behavior still uses the MBCS libraries.

Building on UNIX/Linux

The Uppercase Translation User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

TIRUPDB

TIRURTL—Ultimate Retry Limit Exit

```
long tirurtl ()
```

Source Code

TIRURTL.C

Purpose

TIRURTL lets you specify a maximum value for the TRANSACTION RETRY LIMIT system attribute. This value can never be exceeded by a SET TRANSACTION RETRY LIMIT statement in an action diagram, or by the Default Retry Limit User Exit.

After the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit User Exit, no more retries can occur, and the application fails with a runtime error.

Arguments

None

Return Code

Long containing the retry limit.

Default Behavior

If the Ultimate Retry Limit User Exit is not used, the maximum value of TRANSACTION RETRY LIMIT will be 99 for all target environments. The Ultimate Retry Limit User Exit can be modified to return a value of zero to suppress all retry attempts.

Building on UNIX/Linux

The Ultimate Retry User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

TIRDRTL

TIRUSRID—User ID Exit

```
void TIRUSRID ( char *rp1;
    char *rp2;
    char *filler_parm;
    char *user_id);
```

Source Code

TIRUSRID.C

Purpose

TIRUSRID is used to supply the user's ID to the application. The user ID is one of the parameters passed to the Security Interface Exit (TIRSECR).

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*filler_parm	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*user_id	Output	A pointer to an 8-byte character array into which the user ID can be returned.

Return Code

None

Default Behavior

The default action taken by this module is to call runtime routine DEFUSRID which returns a default user ID, the value of which depends on the platform on which the system is executing.

Building on UNIX/Linux

The User ID User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

TIRSECR

TIRYYX—Date Exit

```
void TIRYYX (
    struct tiryyx_param_block *pb)
```

Source Code

TIRYYX.C

Purpose

TIRYYX is used to process two-digit or yy-style date input and to set the century part using any fixed-window, sliding-window, or other algorithm of choice, when using CA Gen in the standard map generation mode.

Internally, CA Gen handles four digit year dates correctly assuming the user application uses the yyyy edit pattern throughout. If the user interface is designed to accept a two-digit date entry, and defaulting to the current century is not acceptable, use this exit to implement logic to get the required behavior for defaulting the century part of the date.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*pb	Input/Output	A pointer to a tiryyx structure containing the following items:
return_code	Output	A 4-byte character array containing the current year
current_year	Input	A 4-byte character array containing the current year.
edit_year	Input/Output	A 4-byte character array containing the edit year.

Return Code

None

Default Behavior

The default user exit behavior does not perform any processing and returns.

Building on UNIX/Linux

The Date User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

UNIX/Linux Client Middleware User Exits

Common System Utilities - UNIX and Linux User Exits

All supplied Common System Utilities (CSU) user exits are written using the C++ programming language. The following table briefly describes the Common System Utilities Exits:

Common System Utilities: Language: C++		
User Exit Name	**Source Code**	**Description**
CSUGETLIBRARYVERSIONNAME	csuglvn.cxx	Provide a mapping of specified name to version specific name of Shared Libraries.

Details for the preceding user exits follow in a separate section for each.

CSUGETLIBRARYVERSIONNAME—Version Name mapping Exit

```
void CSUGetLibraryVersionName (char *name,
    char *retName,
    long maxRetNameLen )
```

Source Code

CSUGLVN.CXX

Purpose

This exit provides a mapping of specified name to version specific name if one exists, otherwise the specific name value is returned. This mapping is used when shared libraries are dynamically loaded during cooperative processing.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*name	Input	Pointer to a character string containing a null terminated name to be converted to a version name.
*retName	Output	Pointer to a character string to receive the version name.
maxRetNameLen	Input	Long containing the maximum length of the returning version name.

Return Code

None

Default Behavior

The default mapping table is used.

Building on UNIX/Linux

The CSU Library Version Name User Exit is built as part of the shared library libcsuvn.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).

3. Run make /f csuglvn.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

TCP/IP - Windows User Exits

All supplied TCP/IP Transport user exits are written using the C programming language. The following table briefly describes the TCP/IP Exits:

TCP/IP Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_TCP_DPC_DIRSERV_EXIT	citcpclx.c	TCP/IP Directory Services User Exit
CI_TCP_DPC_HANDLECOMM_COMPLETE	citcpclx.c	Verifies that a data has been processed successfully (a valid send/receive has occurred).
CI_TCP_DPC_SETUPCOMM_COMPLETE	citcpclx.c	Verifies that connection to target server is successful.

Details for the preceding user exits follow in a separate section for each.

CI_TCP_DPC_DIRSERV_EXIT—TCPIP DPC Directory Services Exit

```
Void CI_TCP_DPC_DirServ_Exit (char *hostName,
    char *servName,
    char *nextLoc,
    char *trancode,
    char *procName,
    char *modelName);
```

Source Code

CITCPCLX.C

Purpose

The provided sample TCPIP DPC Directory Services exit is an implementation of Transaction routing. Transaction routing is a conceptual process that lets cooperative flow data be routed from a Distributed Process Client (DPC) to a programmatically determined Distributed Process Server (DPS). The supplied sample exit looks for the hostname or IP address and port number or service name in environment variables. The user is free to implement whatever functionality can be required.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*hostName	Input/ Output	The hostname where the target server environment resides according to the configured client.
*servName	Input/ Output	The port number or service name on the target server environment the client is to connect to.
*nextLoc	Input	Next Location system attribute as set using CA Gen action diagram statements.
*trancode	Input	The target Procedure Step transaction code being processed
*procName	Input	The name of the flow's target Procedure Step.
*modelName	Input	The name of the model containing the target Procedure Step.

Return Code

None

Default Behavior

If the environment variables expected by the sample implementation of this exit are not defined the hostname and service name values defined during the packaging of the cooperative model will be used.

Building on UNIX/Linux

The TCP/IP DPC Directory Services User Exit is built as part of the shared library libtcpcx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f ctcpexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

CI_TCP_DPC_HANDLECOMM_COMPLETE—TCP/IP DPC Handle Comm Complete Exit

```
int CI_TCP_DPC_handleComm_Complete(int completionCode,
    int numberOfAttempts,
    unsigned long reasonCode)
```

Source Code

CITCPCLX.C

Purpose

The processing of a given cooperative flow is broken up into two large grain activities. This, the second is handleComm, which is invoked to send/receive data over an already active connection. This exit is invoked at the completion of the handleComm processing to expose that processing results. The handleComm will either be successful (indicated by the input parameter completionCode having a value of HANDLECOMM_OK) or not successful (indicated by the completionCode parameter having a value of HANDLECOMM_NOT_OK).

The return from this exit indicates if the processing of the cooperative flow should continue (zero lets the process continue, non-zero causes the processing of the flow to be terminated).Therefore, if a completionCode of HANDLECOMM_OK is received as input, the return value from this exit should be set to zero to let the processing of the flow continue.

If the completionCode has a value of HANDLECOMM_NOT_OK, this exit has the opportunity to indicate if the handleComm processing should be attempted by returning a value of zero (0).The number of flow attempts is passed into this exit using the numberOfAttempts parameter. Thus, this exit can control the number of retry attempts by testing the value of numberOfAttempts and returning one (1) when the number of retries has reached a predetermined threshold.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
completionCode	Input	An integer value representing the result of a handleComm operation. The value will be either HANDLECOMM_OK or HANDLECOMM_NOT_OK
numberOfAttempts	Input	An integer value representing the number of times a handleComm operation has been attempted. This number will be incremented each time handleComm fails.
reasonCode	Input	An unsigned long value providing the error number as reported by the underlying TCP/IP transport layer. This error code can be used by this exit to determine if a retry is feasible.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
zero (0)	Indicates processing of the flow should continue, retrying the flow processing if not already successful.
non-zero	Causes the processing of the flow to be terminated.

Default Behavior

The default implementation of this exit lets the flow processing be attempted twice. If the flow is not successful after two attempts, the flow processing is terminated and an appropriate error response is returned to the DP client.

Building on UNIX/Linux

The TCP/IP DPC Comm Complete User Exit is built as part of the shared library libtcpcx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f ctcpexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_TCP_DPC_SETUPCOMM_COMPLETE

CI_TCP_DPC_SETUPCOMM_COMPLETE—TCP/IP DPC Setup Comm Complete Exit

```
int CI_TCP_DPC_setupComm_Complete (int completionCode,
    int numberOfAttempts,
    unsigned long reasonCode);
```

Source Code

CITCPCLX.C

Purpose

The processing of a given cooperative flow is broken up into two large grain activities. The first is setupComm, which is invoked to insure a connection to the target server is available. This exit is invoked at the completion of the setupComm processing to expose that processing results. The setupComm will either be successful (indicated by the input parameter completionCode having a value of SETUPCOMM_OK) or not successful (indicated by the completionCode parameter having a value of SETUPCOMM_NOT_OK).

The return from this exit indicates if the processing of the cooperative flow should continue (zero lets the process continue, non-zero causes the processing of the flow to be terminated). If the completionCode has a value of SETUPCOMM_NOT_OK, this exit has the opportunity to indicate if the setupComm processing should be attempted by returning a value of zero(0). The number of connection retries attempted is passed into this exit using the numberOfAttempts parameter. Thus, this exit can control the number of retry attempts by testing the value of numberOfAttempts and returning one (1) when the number of retries has reached a predetermined threshold.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
completionCode	Input	An integer value representing the result of a setupComm operation. The value will be either SETUPCOMM_OK or SETUPCOMM_NOT_OK
numberOfAttempts	Input	An integer value representing the number of times a setupComm has been attempted. This number will be incremented each time setupComm fails.
reasonCode	Input	An unsigned long value providing the error number as reported by the underlying TCP/IP transport layer. This error code can be used by this exit to determine if a retry is feasible.

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
zero (0)	Indicates processing of the flow should continue, retrying the connection if not already established.
non-zero	Causes the processing of the flow to be terminated.

Default Behavior

The default implementation of this exit lets the connection be attempted twice. If the connection is not established after two attempts, the flow processing is terminated and an appropriate error response is returned to the DP client.

Building on UNIX/Linux

The DPC Setup Comm Complete User Exit is built as part of the shared library libtcpcx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f ctcpexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_TCP_DPC_HANDLECOMM_COMPLETE

WebSphere MQ Client Transport - Windows User Exits

All supplied WebSphere MQ Client Transport user exits are written using the C programming language. The following table briefly describes the WebSphere MQ Exits:

WebSphere MQ Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_MQS_DPC_EXIT	cimqclex.c	MQ Directory Services Exit
CI_MQS_DPC_HANDLECOMM_COMPLETE	cimqclex.c	Verifies that a data has been processed successfully (a valid send/receive has occurred).
CI_MQS_DPC_SETREPORTOPTIONS	cimqclex.c	Used to override report options set by the runtime.
CI_MQS_DPC_SETUPCOMM_COMPLETE	cimqclex.c	Verifies that connection to target server is successful.

WebSphere MQ Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_MQS_DYNAMICQNAME_EXIT	cimqclex.c	Provide Queue Name that will be used when opening a dynamic queue.
CI_MQS_MQSHUTDOWNTEST	cimqclex.c	Determine if queue should be removed and thus disconnected.

Details for the preceding user exits follow in a separate section for each.

CI_MQS_DPC_EXIT—MQSeries DPC Directory Services Exit

```
Void CI_MQS_DPC_Exit (char *qMgr,
    char *rqMgr,
    char *pQ,
    char *rQ,
    long *timeout,
    short *closePQ,
    short *closeGQ,
    char *nextLoc,
    char *trancode,
    char *procName,
    char *modelName);
```

Source Code

CIMQCLEX.C

Purpose

The provided sample WebSphere MQ DPC Directory Services exit is an implementation of Transaction routing. Transaction routing is a conceptual process that lets cooperative flow data be routed from a Distributed Process Client (DPC) to a programmatically determined Distributed Process Server (DPS). The current cooperative request's local queue manager, remote queue manager, put and reply queue names can be overridden using this exit. Additionally a get timeout value as well as put/get queue disposition after a flow has completed, can be customized.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*qMgr	Input/Output	The name of the local queue manager. By default, the application obtains this information from the model during generation. This exit can override this name.
*rqMgr	Input/Output	The name of the remote queue manager. This value is NULL by default. This exit can override this name.
*pQ	Input/Output	A character string containing the name of the Put queue. By default, the application obtains this information from the model during generation. This exit can override this name.
*rQ	Input/Output	A character string containing the name of the reply-to queue. This can be either a local queue or a model queue name. By default, the application obtains the value "SYSTEM.DEFAULT.MODEL.QUEUE" from the model during generation. This exit can override this name.
*timeout	Input/Output	A long value representing the timeout value, in milliseconds, for the Get queue. By default, this has the value MQWI_UNLIMITED for an unlimited waiting period. This exit can override this name.
*closePQ	Input/Output	A short value which controls whether the client closes the Put queue after the flow is complete. Valid values are CLOSE_QUEUE or NO_CLOSE_QUEUE. The default value, NO_CLOSE_QUEUE, specifies the Put queue is not to be closed. This exit can override this name.
*closeGQ	Input/Output	A short value which controls whether the client closes the Get queue after the flow is complete. Valid values are CLOSE_QUEUE or NO_CLOSE_QUEUE. The default value, NO_CLOSE_QUEUE, specifies the Get queue is not to be closed. This exit can override this name.
*nextLoc	Input	A character string containing the Next Location system attribute as set using CA Gen action diagram statements.
*trancode	Input	An 8-byte character array containing the target Procedure Step transaction code being processed.
*procName	Input	A character string containing the name of the flow's target Procedure Step.

Name	I/O	Description
*modelName	Input	A character string containing the name of the model containing the flow's target Procedure Step.

Return Code

None

Default Behavior

The supplied sample does not implement dynamic transaction routing. For more information about default values for the various parameters, see Arguments.

Building on UNIX/Linux

The Websphere MQ DPC Directory Services User Exit is built as part of the shared library libmqscx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f cmqsexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

CI_MQS_DPC_HANDLECOMM_COMPLETE—Handle Comm Retry Count Exit

```
int CI_MQS_DPC_handleComm_Complete(int completionCode,
    int numberOfAttempts,
    unsigned long reasonCode);
```

Source Code

CIMQCLEX.C

Purpose

The processing of a given cooperative flow is broken up into two large grain activities. The second is handleComm, which is invoked to send/receive data over an already active connection. This exit is invoked at the completion of the handleComm processing to expose that processing results. The handleComm will either be successful (indicated by the input parameter completionCode having a value of HANDLECOMM_OK) or not successful (indicated by the completionCode parameter having a value of HANDLECOMM_NOT_OK).

The return from this exit indicates if the processing of the cooperative flow should continue (zero lets the process continue, non-zero causes the processing of the flow to be terminated). Therefore, if a completion code of HANDLECOMM_OK is received as input, the return value from this exit should be set to zero to let the processing of the flow continue.

If the completionCode has a value of HANDLECOMM_NOT_OK, this exit has the opportunity to indicate if the handleComm processing should be attempted by returning a value of zero (0). The number of flow attempts is passed into this exit using the numberOfAttempts parameter. Thus, this exit can control the number of retry attempts by testing the value of numberOfAttempts and returning one (1) when the number of retries has reached a predetermined threshold.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
completionCode	Input	An integer value representing the result of a handleComm operation. The value will be either HANDLECOMM_OK or HANDLECOMM_NOT_OK
numberOfAttempts	Input	An integer value representing the number of times a handleComm operation has been attempted. This number will be incremented each time handleComm fails.
reasonCode	Input	An unsigned long value providing the error number as reported by the underlying WebSphere MQ transport layer. This error code can be used by this exit to determine if a retry is feasible.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
zero (0)	Indicates processing of the flow should continue, retrying the flow processing if not already successful.
non-zero	Causes the processing of the flow to be terminated.

Default Behavior

The default implementation of this exit lets the flow processing be attempted twice. If the flow is not successful after two attempts, the flow processing is terminated and an appropriate error response is returned to the DP client.

Building on UNIX/Linux

The Websphere MQ DPC Handle Comm Retry Count User Exit is built as part of the shared library libmqscx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f cmqsexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_MQS_DPC_SETUPCOMM_COMPLETE

CI_MQS_DPC_SETREPORTOPTIONS—Override Put Queue Report Options Exit Description

```
void CI_MQS_DPC_setReportOptions( MQLONG * reportOptions);
```

Source Code

CIMQCLEX.C

Purpose

This exit can be used to override the set of report options defined for the WebSphere MQ Put Message Descriptor prior to the issuance of an MQPUT() operation.

The report options set by the runtime are described in the following table:

Report Option	Description
MQRO_EXCEPTION	This type of report can be generated when an exception occurs. For instance if a message is sent to another queue manager and the message cannot be delivered to the specified destination queue.
MQRO_EXPIRATION	An expiration report. The queue manager generates this type of report if the message is discarded prior to delivery to an application because its expiry time has passed.
MQRO_PASS_MSG_ID	If a report is generated, the MsgId of the current message being processed is to be copied to the MsgId of the report message.
MQRO_COPY_MSG_ID_TO_CORREL_ID	Indicates the correlation ID of the report generated should equal the message ID of the request originally issued.
MQRO_DEAD_LETTER_Q	This option causes the original message to be placed on the dead-letter queue when an exception occurs

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*reportOptions	Input/ Output	A pointer to a long value representing the currently defined report options to be used in the Put Message Descriptor.

Return Code

None

Default Behavior

The runtime specified report options are left unchanged.

Building on UNIX/Linux

The Websphere MQ Override Put Queue Report Options User Exit is built as part of the shared library libmqscx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f cmqsexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

CI_MQS_DPC_SETUPCOMM_COMPLETE—Setup Comm Retry Count Exit

```
int CI_MQS_DPC_setupComm_Complete(int completionCode,
    int numberOfAttempts,
    unsigned long reasonCode);
```

Source Code

CIMQCLEX.C

Purpose

The processing of a given cooperative flow is broken up into two large grain activities. The first is setupComm, which is invoked to insure a connection to the target server is available. This exit is invoked at the completion of the setupComm processing to expose that processing results. The setupComm will either be successful (indicated by the input parameter completionCode having a value of SETUPCOMM_OK) or not successful (indicated by the completionCode parameter having a value of SETUPCOMM_NOT_OK).

The return from this exit indicates if the processing of the cooperative flow should continue (zero lets the process continue, non-zero causes the processing of the flow to be terminated). If the completionCode has a value of SETUPCOMM_NOT_OK, this exit has the opportunity to indicate if the setupComm processing should be attempted by returning a value of zero (0). The number of connection retries attempted is passed into this exit using the numberOfAttempts parameter. Thus, this exit can control the number of retry attempts by testing the value of numberOfAttempts and returning one (1) when the number of retries has reached a predetermined threshold.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
completionCode	Input	An integer value representing the result of a setupComm operation. The value will be either SETUPCOMM_OK or SETUPCOMM_NOT_OK.
numberOfAttempts	Input	An integer value representing the number of times a setupComm has been attempted. This number will be incremented each time setupComm fails.
reasonCode	Input	An unsigned long value providing the error number as reported by the underlying WebSphere MQ transport layer. This error code can be used by this exit to determine if a retry is feasible.

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
zero (0)	Indicates processing of the flow should continue, retrying the connection if not already established.
non-zero	Causes the processing of the flow to be terminated.

Default Behavior

The default implementation of this exit lets the connection be attempted twice. If the connection is not established after two attempts, the flow processing is terminated and an appropriate error response is returned to the DP client.

Building on UNIX/Linux

The Websphere MQ DPC Setup Comm Retry Count User Exit is built as part of the shared library libmqscx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f cmqsexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_MQS_DPC_HANDLECOMM_COMPLETE

CI_MQS_DYNAMICQNAME_EXIT—Dynamic Queue Name Override Exit

```
void CI_MQS_DynamicQName_Exit (char *dynamicQName);
```

Source Code

CIMQCLEX.C

Purpose

The Dynamic Queue Name exit lets you override the queue name that will be used when opening a dynamic queue. The resulting Dynamic Queue will obtain its attributes from the specified WebSphere MQ Model Queue name. The passed in work area can be modified by placing a null terminated string of the value to be used as the dynamic queue name, including the use of valid WebSphere MQ pattern characters used to name dynamic queues.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*dynamicQName	Input/ Output	A pointer to a character buffer, of length MQ_Q_NAME_LENGTH+1 (48 +1), that contains the default name of the dynamic queue as built by the WebSphere MQ runtime (for example username.processid.threadid).

Return Code

None

Default Behavior

The dynamic queue name is not modified.

Building on UNIX/Linux

The Websphere MQ Dynamic Queue Name Override User Exit is built as part of the shared library libmqscx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f cmqsexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

CI_MQS_MQSHUTDOWNTEST—MQSeries Queue Disconnect Exit

```
Int CI_MQS_MQShutdownTest()
```

Source Code

CIMQCLEX.C

Purpose

This exit can be used to modify the behavior of the normal put/get queue disposition after successful completion of a cooperative flow. Normal disposition will leave the connection valid with the put and get queues open, ready to handle subsequent flows. This exit can override that behavior and cause the queues and connection to be closed after each flow has completed.

Arguments

None

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
NO_REMOVE_QUEUE	The default return value. The connection and put/get queues will remain open, available for subsequent flows.
REMOVE_QUEUE	After a completed flow the connection is dropped, the put/get queues will be closed.

Default Behavior

A value of NO_REMOVE_QUEUE is returned, the connection and put/get queues remain open and available for subsequent flows.

Building on UNIX/Linux

The Websphere MQ Queue Disconnect User Exit is built as part of the shared library libmqscx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f cmqsexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

Tuxedo

All supplied Tuxedo Transport user exits are written using the C programming language. The following table briefly describes the Tuxedo Exits:

Tuxedo Transport: Language: C

User Exit Name	Source Code	Description
CI_C_SEC_SET	cictuxwsx.c	Set User Supplied Security Data Into The Security Data Fields Located In Tuxedo Tpinit
CI_C_USER_DATA_IN	cictuxwsx.c	Gives You The Opportunity To Inspect Or Modify The Cooperative Flow Request Buffer On Return From The Target Tuxedo Service. Invoked On Return From The Tpcall For Those Clients Connecting To Tuxedo Servers Residing On A Separate Host. Additionally, This Exit Allows The Client To Disconnect From The Server Following Each Flow

Tuxedo Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_C_USER_DATA_IN	cictuxx.c	Gives You The Opportunity To Inspect Or Modify The Cooperative Flow Request Buffer On Return From The Target Tuxedo Service. Invoked On Return From The Tpcall For Those Clients Connecting To Tuxedo Servers Residing On A Separate Host. Additionally, This Exit Allows The Client To Disconnect From The Server Following Each Flow (used For Server To Server Flows)
CI_C_USER_DATA_OUT	cictuxwsx.c	Gives You The Opportunity To Inspect Or Modify The Cooperative Flow Request Buffer Prior To Tuxedo Sending The Request To The Target Tuxedo Service. Invoked Prior To The Tpcall For Those Clients Connecting To Tuxedo Servers Residing On A Separate Host
CI_C_USER_DATA_OUT	cictuxx.c	Gives You The Opportunity To Inspect Or Modify The Cooperative Flow Request Buffer Prior To Tuxedo Sending The Request To The Target Tuxedo Service. Invoked Prior To The Tpcall For Those Clients Connecting To Tuxedo Servers Residing On A Separate Host (used For Server To Server Flows)
CI_EVENT_HANDLER	cictuxwsx.c	Provides ability to handle events

Note that both cictuxwsx.c and cictuxx.c have essentially the same user exits. The difference is that the user exits in cictuxwsx.c are used for client/server flows, while the user exits in cictuxx.c are used for server to server flows.

Details for the preceding user exits follow in a separate section for each.

CI_C_SEC_SET—Tuxedo Cooperative Flow Security Exit

```
int ci_c_sec_set (CIPROCSTEP *procstep,
    char *clientUserid,
    char *clientPassword,
    int *tperr )
```

Source Code

CICTUXWSX.C

Purpose

The ci_c_sec_set exit is the first user exit that is invoked on a client Tuxedo cooperative flow, and is specifically related to security. ci_c_sec_set() contains the call to tpinit() and is called for each flow so that, if required, each flow can establish a different user context (that is, tpinit() is invoked for each flow). You need to invoke the corresponding tpterm in ci_c_user_data_in() to achieve the described behavior.

ci_c_sec_set() is called from a Windows Client only. A Tuxedo server-to-server flow does not invoke ci_c_sec_set().

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*procstep	Input	A pointer to a CA Gen CIPROCSTEP structure, which contains information, related to the currently executing procedure step. Model name and Next Location data can be extracted from this structure. This parameter is unused in the supplied sample exit.
*clientUserid	Input	CA Gen current CLIENT_USER_ID system attribute, if SecurityUsedEnhanced is set in the WRSECTOKEN user exit. NULL string otherwise.
*clientPassword	Input	CA Gen current CLIENT_PASSWORD system attribute if SecurityUsedEnhanced is set in the WRSECTOKEN user exit. NULL string otherwise.
*tperr	Output	If the function returns -1 as the return code, tperr should contain a valid tperrno value indicating the cause of the Tuxedo ATMI call failure.

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
zero (0)	If no error was encountered within the exit.
non-zero	If one of the ATMI functions within user, the exit fails.

Default Behavior

The clientUserId and clientPassword are used for user name and the data field of the TPINIT structure respectively. TPINIT structure is used for the Tuxedo login. If SecurityUsedEnhanced is returned from the WRSECTOKEN user exit, clientUserid and clientPassword contain a pointer to a valid CA Gen CLIENT_USER_ID and CLIENT_PASSWORD system attributes, which are used for the Tuxedo login. Otherwise, the pointers point to a Null string.

Building on UNIX/Linux

The Tuxedo Cooperative Flow Security User Exit is built as part of the shared library libtxwcx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f ctuxexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_EVENT_HANDLER

CI_C_USER_DATA_IN—Tuxedo Inbound Flow Data Access Exit

```
void  ci_c_user_data_in(char ** tuxSvcOutputBuffer)
```

Source Code

CICTUXWSX.C, CICTUXX.C

Purpose

The ci_c_user_data_in exit is called immediately after the Tuxedo tpcall API returns. It provides access to the inbound View32 buffer on returning from the target server. tuxSvcOutputBuffer points to the reply buffer, which contains a CA Gen procedure, step structure and export view (back to back).

If required, a call to the Tuxedp tpterm API can be added inside ci_c_user_data_in to force association for each flow with a discrete user environment and privileges (see ci_c_sec_set).

ci_c_user_data_in() is called from both a Windows Client and Server to Server flows, hence the two source files.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
**tuxSvcOutputBuffer	Input	A pointer to a buffer containing a Tuxedo allocated buffer, which contains procedure step and view data for the flow in progress. This is the buffer returned back from the Tuxedo tpcall() API (inbound View32).

Return Code

None

Default Behavior

The default behavior is to not modify the received data.

Building on UNIX/Linux

The Tuxedo Inbound Flow Data Access User Exit is built as part of the shared library libtxwcx.xx.*, as well as the shared library libtxcx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).

3. Run make /f ctuxexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_C_USER_DATA_OUT—Tuxedo Outbound Flow Data Access Exit

CI_C_USER_DATA_OUT—Tuxedo Outbound Flow Data Access Exit

```
void  ci_c_user_data_out(char ** tuxSvcInputBuffer , long * svcFlags );
```

Source Code

CICTUXWSX.C, CICTUXX.C

Purpose

The ci_c_user_data_out() function is the second user exit that is invoked on a cooperative flow. It provides access to the outbound View32 buffer, and an opportunity to add more attributes to the flags parameter of the subsequent Tuxedo tpcall API. tuxSvcInputBuffer is a request buffer containing a CA Gen procedure step structure followed by its import views as back-to-back data items. svcFlags are added to the tpcall flags parameter. Both are passed to the tpcall.

ci_c_user_data_out() is called from both a Windows Client and Server to Server flows, hence the two source files.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
**tuxSvcInputBuffer	Input/ Output	A pointer to a buffer containing a Tuxedo allocated buffer, which contains procedure step and view data for the flow in progress.
*svcFlags	Output	A pointer to a long value to which can be stored extra flags to be appended to the standard value passed as the flags parameter to the Tuxedo tpcall() API.

Return Code

None

Default Behavior

The flow data and flags parameter passed to tpcall() are not modified.

Building on UNIX/Linux

The Tuxedo Outbound Flow Data Access User Exit is built as part of the shared library libtxwcx.xx.*, as well as the shared library libtxcx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f ctuxexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_C_USER_DATA_IN—Tuxedo Inbound Flow Data Access Exit

CI_EVENT_HANDLER—Tuxedo Event Handler Exit

void TUXCALL ci_event_handler(char * s, long len, long flag);

Source Code

CICTUXWSX.C

Purpose

The ci_event_handler() function is called in ci_c_sec_set() to respond to the runtime failure from the call to TPINIT().

ci_event_handler() is called from a Windows Client only. A Tuxedo server-to-server flow does not invoke ci_c_sec_set() and therefore does not invoke ci_event_handler().

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*s	Input	A pointer to a buffer containing a Tuxedo error message.
Len	Input	A long value containing the length of the error message.
Flag	Input	A long value containing an error flag.

Return Code

None

Default Behavior

CI_EVENT_HANDLER does no work.

Building on UNIX/Linux

The Tuxedo Event Handler User Exit is built as part of the shared library libtxwcx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).

3. Run make /f ctuxexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_C_SEC_SET

UNIX and Linux Server User Exits

The following table summarizes the functions available through the user exits for generated server applications:

Name	Description
DBCOMMIT	Database Commit User Exit. There is one user exit routine for each supported database: Oracle, and DB2.
DBCONNCT	Database Connection User Exit. There Is One User Exit Routine For Each Supported Database: Oracle, And Db2.
DBDISCNT	Database Disconnect User Exit. There is one user exit routine for each supported database: Oracle, and DB2.
SRVRERROR	Server to Server Error User Exit (Server Only)
TIRDCRYP	Decrypt User Exit (Server Only)
TIRDLCT	Dialect User Exit
TIRDRTL	Default Retry Limit User Exit
TIRELOG	Server Error Logging User Exit (Server Only)
TIRHELP	Help Interface User Exit
TIRMTQB	Message Table User Exit
TIRNCRYP	Encrypt User Exit (Server Only)
TIRSECR	Security Interface User Exit
TIRSECV	Server Security Validation User Exit (Server Only)
TIRSYSID	System ID User Exit
TIRTERMA	User Termination User Exit

Name	Description
TIRTIAR	Database error message User Exit. There is one user exit routine for each supported database: Oracle and DB2.
TIRUPDB	MBCS Uppercase Translation User Exit
TIRUPPR	Uppercase Translation User Exit
TIRURTL	Ultimate Retry Limit User Exit
TIRUSRID	User ID User Exit
TIRXINFO	Locale Information User Exit (Server Only)
TIRXLAT	National Language Translation User Exit (Server Only)
TIRYYX	Date User Exit

Note: The database user exits DBCONNCT, DBCOMMIT, DBDISCNT and TIRTIAR are rebuilt into individual shared libraries (libae_db2.*, libae_ora.*) using the script $IEFH/make/mkdbs.

Server runtime user exits are rebuilt into the shared library libae_userexits_c.* using the script $IEFH/make/mkexits. This is the same shared library that is used with Blockmode applications.

Since a large number of these user exits have already been documented in the section UNIX/Linux Blockmode User Exits, only the user exits that are specific to server applications will be detailed in the following subsections.

Details for the preceding user exits follow in a separate section for each.

SRVRERROR—Server to Server Error Exit

```
int SRVRERROR ( char * from,
    char * to,
    char * errLst,
    int dtp,
    int failureType,
    char * failureCommand,
    ErrorToken errorToken);
```

Source Code

tirserrx.c

Purpose

This exit is invoked by the calling server when errors occur at the destination server, during a server-to-server flow. This exit can influence the default runtime error behavior in how the detected error is handled. When the NOTPROPAGATE_ERR is returned, the calling procedure step continues the execution, ignoring the fact that an error has occurred in the destination procedure step. When PROPAGATE_ERR is returned, an error message is created and then returned to the calling procedure step.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*from	Input	A character string pointing to the name of the source or calling procedure step
*to	Input	A character string pointing to the name of the target or called procedure step
*errLst	Input	A character string pointing to the destination server's XFAIL message. This message buffer can contain multiple new line terminated strings. The initial portion of the buffer is formatted to fit a 24 line by 80-character screen format. Additional free form data can follow the 24 x 80 lines, up to the maximum of 2048 bytes.
dtp	Input	An integer value of 1 if the to procedure step is a distributed transaction participant, 0 if otherwise
failureType	Input	An enumerated value representing a failure code CFBUILD = 0 - implies the calling procedure step failed to build/parse the message bound for the destination procedure step XFAIL = 1 - implies the destination procedure step execution failed XERR = 2 - implies a communication error between the calling and destination procedure steps
*failureCommand	Input/Output	A command that the destination procedure step can return to the calling procedure step. A maximum of 8 chars plus NULL.
errorToken	Input/Output	This parameter is only used with XFAL messages. errorToken can contain a token constructed by the Error Logging exit (TIRELOG) at the target or called procedure step. (4096 +1 bytes)

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
0 - PROPAGATE_ERR	The error is processed normally within the calling procedure step.
1-NOTPROPAGATE_ERR	The calling procedure step continues the execution ignoring the fact that error occurs in the destination procedure step.

Default Behavior

Errors are propagated to the calling procedure step. An error message is created and returned to the calling procedure step.

Building on UNIX/Linux

This user exit is built as part of the shared library or archive library libae_userexits_c.*, where * is the shared library suffix or archive library depending on the UNIX system. As a prerequisite for building the shared library or archive library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f stuxexit.plat, where plat is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

TIRDCRYP—Server Decryption Exit

```
void TIRDCRYP( unsigned char * rp1,
    unsigned char * rp2,
    TIRDCRYP_cmcb * pTIRDCRYP_cmcb);
```

Source Code

tirdcryp.c

Purpose

TIRDCRYP is called by the Server Manager after it detects that the client has sent an encrypted cooperative buffer. The Server Manager constructs a work buffer containing the concatenated View Data and Client Security sections. The user is responsible for decrypting the area pointed to by pDataBuffer for IBufferSize bytes.

The inputs pDataBuffer and IDecryptMaxSize as well as the outputs IBufferSize return_code and failure_msg are fields within a structure pointed to by the pTIRDCRYP_cmcb parameter.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only
*pTIRELOG_cmcb	Input/Output	A pointer to a TIRELOG_CMCB structure containing the following items:
lDecryptMaxSize	Input	A long field that contains the maximum available buffer space (in bytes) that the decrypted data can occupy.
lBufferSize	Input/Output	On input, IBufferSize is the current buffer space (in bytes) of the encrypted data. On output, IBuffferSize should be updated by this exit to contain the length of the decrypted data. The length of the decrypted result cannot exceed lDecryptMaxSize.

Name	I/O	Description
*pDataBuffer	Input/Output	On input, a pointer to the starting location of the encrypted View Data and Client Security sections within the CFB work buffer. On output, this exit should ensure this same data area contains the unencrypted versions of the input data. The length of this decrypted result cannot exceed lDecryptMaxSize.
return_code	Output	A two-character array returning the results of the decryption attempt. The following values are supported: DECRYPTION_USED—defined as " " DECRYPTION_SIZE_EXCEEDED_MAX—defined as "01" DECRYPTION_NOT_USED—defined as "02" DECRYPTION_APPLICATION_ERROR—defined as "03"
*failureMsg	Output	The pointer to an 80-character array, to be populated by the exit that can receive a null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of DECRYPTION_APPLICATION_ERROR.

Return Code

None directly, see the preceding return_code structure member.

Default Behavior

Decryption of the data buffer is not attempted.

Building on UNIX/Linux

The Server Decryption User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

The following are related user exits:

- TIRNCRYP
- WRSECENCRYPT

TIRELOG—Server Error Logging and Error Token Creation Exit

```
void TIRELOG(char * rp1,
    char * rp2,
    TIRELOG_CMCB * pTIRELOG_cmcb));
```

Source Code

tirelog.c

Purpose

This exit serves two purposes:

- Error logging at the server
- Creation of an error token for transmitting to the client

This exit is called by the server to handle server errors that are encountered during the execution of a distributed processing server that cannot be handled by the runtime or generated code, and normally result in the termination of the application. For example, prior to the execution of a server procedure step, the server extracts view data from the client message and places it in the target procedure step's view. If this extraction fails because of a mismatch between the client definition and the server definition an error response message is created and returned to the client.

The default implementation of this exit returns to the caller without logging the error. It is up to the developer of this user exit to determine what information should be logged and how it should be logged. Some users can choose to log only certain errors; others can choose to log all errors. On some systems, the log can be implemented as a file. To log a server error, simply format the information you wish to log and write it to a file. On other systems, the log can be implemented using system-specific features such as a CICS temporary storage queue (TSQ) as found on z/OS.

To create an error token, move text data to the area pointed to by the elog_error_token member of the TIRELOG_CMCB structure passed into this exit. The error token area is 4097 bytes and must be null-terminated. The error token, which goes through codepage translation when it is transmitted to the client, can be used on the client to customize how the error is handled.

For example, you can modify this exit to return an error token of "RETRY" whenever a certain database contention error occurs. This error token is passed to the client error-handling exit (WRSRVRERROR or WRASYNCSRVRERROR), which makes the final decision on how to handle the error. You can modify the client error-handling exit to reinvoke the flow or USE whenever the error token is "RETRY." This server error-logging exit is called after the error response message is created but before it is transmitted to the client.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only
*pTIRELOG_cmcb	Input/Output	A pointer to a TIRELOG_CMCB structure containing the following items:
elog_fail_type	Input	A character designating the type of failure detected defined as:(lable - defined value) EPROFD - 'P' profile error EPROFI - 'I' profile error EEXEC - 'E' execution error ESERVER - 'D' server manager error EUSER - 'U' user requested abend
void *elog_sqlca	Input	A pointer to a saved sql data area
*elog_globdata	Input	A pointer to the server's globdata area
elog_number_of_lines	Input	An integer containing the number of text lines contained within the elog_error_text buffer
elog_error_text	Input	A pointer to a buffer of screen formatted text. This data, formatted by the server runtime, contains up to 24 lines of 80 characters each.
*elog_error_token	Input/Output	A character pointer to an error token area that can contain up to 4097 bytes, this includes the required null terminator. This exit is responsible for populating this data area if needed.

Return Code

None directly, see the preceding pTIRELOG_cmcb structure.

Default Behavior

The default action is to return without logging the error.

Building on UNIX/Linux

The Server Error Logging User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

The following are related user exits:

- WRSRVRERROR
- WRASYNCSRVRERROR

TIRNCRYP—Server Encryption Exit

```
void TIRNCRYP( unsigned char * rp1,
    unsigned char * rp2,
    TIRNCRYP_cmcb * pTIRNCRYP_cmcb);
```

Source Code

tirncryp.c

Purpose

After a server procedure step executes, the server manager can call TIRNCRYP to encrypt the server response to the client. The server manager makes a copy of the unencrypted cooperative buffer pending transmission back to the client. The inputs pDataBuffer, IBufferSize, IEncryptMaxSize trancode and client_userid as well as the outputs return_code, and failure_msg are fields with a structure pointed to by pTIRNCRYP_cmcb. The user is responsible for encrypting the data area pointed to by the pDataBuffer member of the TIRNCRYP_cmcb structure.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*pTIRNCRYP_cmcb	Input/Output	A pointer to a structure containing the following items:
pDataBuffer	Input/Output	On input, a pointer to the starting location of the View Data and Client Security sections within the CFB work buffer. On output this same data area should be populate by this exit with the encrypted versions of the input data. The length of the encrypted result cannot exceed lEncryptMaxSize.
lBufferSize	Input/Output	On input, lBufferSize is the current buffer space (in bytes) of the unencrypted data. On output, lBuffferSize should be updated by this exit to contain the length of the encrypted data. The length of the encrypted result cannot exceed lEncryptMaxSize.
lEncryptMaxSize	Input	A long field that contains the maximum available buffer space (in bytes) that the encrypted data can occupy.
trancode	Input	Transaction code currently being processed. . This value can be used in conjunction with client userid and NextLocation to determine if encryption is desired.
client_userid	Input	Client user ID. This value can be used in conjunction with trancode and NextLocation to determine if encryption is desired.
pNextLocation	Input	Next Location value as set by the server application using CA Gen action diagram statements. This value can be used in conjunction with trancode and client userid to determine if encryption is desired.
return_code	Output	A two-character array returning the results of the decryption attempt. The following values are supported: ENCRYPTION_USED—defined as " " ENCRYPTION_SIZE_EXCEEDED_MAX—defined as "01" ENCRYPTION_NOT_USED—defined as "02" EnCRYPTION_APPLICATION_ERROR—defined as "03"

Name	I/O	Description
*failureMsg	Output	The pointer to an 80-character array, to be populated by the exit that can receive a null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of ENCRYPTION_APPLICATION_ERROR.

Return Code

None directly, see the preceding return_code structure member.

Default Behavior

The default logic of this user exit is to return ENCRYPTION_NOT_USED.

Building on UNIX/Linux

The Server Encryption User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

The following are related user exits:

- TIRDCRYP
- WRSECDECRYPT

TIRSECV—Security Validation Exit

```
void TIRSECV(char *rp1,
    char *rp2,
    unsigned char Enhanced_Security_Flag,
    PTIRSECV_cmcb pTIRSECV_cmcb);
```

Source Code

tirsecv.c

Purpose

This security exit is called for every cooperative flow, regardless of the security type used. To facilitate security validation a flag indicating whether the security data is for a standard or enhanced buffer has been added. This exit is intended to provide the opportunity to validate enhanced security data while at the same time not impacting those using standard security.

To this effect, the default code provided handles two possible conditions:

- For buffers containing standard security the client userid, client password, and security token fields are expected to be blank. The default behavior is for the exit to return SECURITY_USED, thus indicating that the request is authorized. The exit must be modified to return SECURITY_APPLICATION_ERROR if the intent is that all buffers contain enhanced security data.
- For buffers containing enhanced security the client userid, client password, and security token fields can or cannot contain data. The default behavior is for the exit to return SECURITY_NOT_USED, this indicating that no validation processing was attempted. The exit must be modified to validate the security data and set the relevant return code (return SECURITY_USED for an authorized user and SECURITY_APPLICATION_ERROR for a non authorized user). When returning SECURITY_APPLICATION_ERROR, this exit can provide an optional failure message, using the failure_msgbuffer contained within the TIRSECV_cmcb structure that will be presented to the client.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
Enhanced_Security_Flag	Input	A single character denoting if the CFB has been created to support enhanced security. A value of Y denotes enhanced security,
*pTIRSECV_cmcb	Input	A pointer to a structure containing the following values:

Name	I/O	Description
client_userid	Input	A 64-byte character array containing a user ID if the CFB uses enhanced security. For a CFB containing standard security this parameter is expected to be blank.
client_password	Input	A 64-byte character array containing a password if the CFB uses enhanced security. For a CFB containing standard security this parameter is expected to be blank.
lSecurityTokenLen	Input	A long value representing the length of the pSecurityToken, if any.
pSecurityToken	Input	A pointer to a security token if the CFB uses enhanced security. For a CFB containing standard security this parameter is expected to be blank.
trancode	Input	An 8-byte character array containing the transaction code
return_code	Input/Output	A 2-byte character array containing a value denoting success for failure of this exit. Valid values are: SECURITY_USED - defined as " " SECURITY_NOT_USED—defined as "02" SECURITY_APPLICATION_ERROR—defined as "03"
failure_msg	Input/Output	The pointer to an 80-character array that can be populated by this exit with a null terminated error message string. The string pointed to by this parameter will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of SECURITY_APPLICATION_ERROR.

Return Code

None directly, see the preceding return_code structure member.

Default Behavior

The default logic of this user exit is to return SECURITY_NOT_USED, which is considered an error if this user exit is actually called since the Server Manager requested Client Security validation.

Building on UNIX/Linux

The Security Validation User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

WRSECTOKEN

TIRXINFO—Locale Information Exit

```
void TIRXINFO (char *osId,
    char *codePage,
    long *padChar);
```

Source Code

tirxlat.c

Purpose

This exit provides information about the codepage environment of the executing server process. An osId, codepage ID, and default padding character are returned. The runtime uses the osId and codePage returned as parameters passed into the TIRXLAT user exit.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*osId	Output	A pointer to character buffer to contain an OS ID (9 bytes, 8 characters plus NULL terminator). This value will be passed to TIRXLAT as the outOS parameter for inbound transactions and as the inOS parameter for outbound transactions. The current default value is MBCS. This should not be confused with an identifier of the underlying operating system on which the server is executing.
*codePage	Output	A pointer to character buffer to contain a codepage ID (9 bytes, 8 characters plus NULL terminator). This value will be passed to TIRXLAT as the outCodePage parameter for inbound transactions and as the inCodePage parameter for outbound transactions. The default value, as returned from this exit, is hard coded into the generated server manager at code generation time. This value will depend upon the platform used to generate the server manager. If the server manager is generated on a Windows platform the value will be 1252, if generated on a UNIX platform using CSE its value will be 819.
*padChar	Output	A pointer to a long value, not currently used for Windows or UNIX servers.

Return Code

None

Default Behavior

The string returned for osId is currently hard coded to a value of MBCS. The value for CodePage is obtained from the server manager. The CodePage number is created during the server manager code generation process. The padChar value is currently unused.

Building on UNIX/Linux

The Locale Information User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

TIRXLAT

TIRXLAT—National Language Translation Exit

```
void TIRXLAT (char *inBuf;
    long *inLen;
    char *inCodePage;
    char *inOS;
    char *outBuf;
    long *outLen;
    char *outCodePage;
    char *outOS;
    long *outPadChar;
    char *workArea;
    long *outCharCnt;
    long *outByteCnt);
```

Source Code

tirxlat.c

Purpose

TIRXLAT allows the conversion of textual data based on from/to codepage and operating system information. View data that is passed between the client and server is translated from the client's code page to the server's code page, and vice versa. TIRXLAT uses the client's code page value, which is passed from the client to the server, and the host's code page value to locate a translation table.

This exit is used to translating both the data received from that client and the data to be sent to the client.

When translating data received from the client, the in* parameters correspond to the client data, the out* parameters correspond to the data presented to the server.

When translating data to be sent to the client the in* parameters correspond to the server data to be sent, the out* parameters correspond to the data presented to the client.

If a suitable translation table is not found, the data will be passed back without translation. The user can replace a translation table to customize their environment.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*inBuf	Input	A character pointer to the input buffer to translate
*inLen	Input	A pointer to a long value which is the length inBuf
*inCodePage	Input	A character pointer to the codepage ID of inBuf (8 bytes + 1 NULL).
*inOS	Input	A character pointer to the OS ID of inBuf (8 bytes + 1 NULL).
*outBuf	Input/Output	A character pointer to the buffer in which to place the translated text
*outLen	Input	A pointer to a long value that is the length of the data pointed to by outBuf.
*outCodePage	Input	A character pointer to the codepage ID corresponding to the output buffer, outBuf.
*outOS	Input	A character pointer to the OS ID corresponding to the output buffer, outBuf.

Name	I/O	Description
*outPadChar	Input	A pointer to a long value which is the padding character to use, 0 if no padding to be done in the output buffer, outBuf.
*workArea	Input	A character pointer to a 100-byte scratch work area.
*outCharCnt	Input/Output	A pointer to a long value which is the number of characters placed into the output buffer, outBuf.
*outByteCnt	Input/Output	A pointer to a long value which is the number of bytes placed into the output buffer, outBuf.

Return Code

None

Default Behavior

If a suitable translation table is found, the data will be translated from the inCodePage to the outCodePage. If a suitable translate table is not found the data is passed back without translation.

Building on UNIX/Linux

The National Language Translation User Exit is built as part of the shared library libae_userexits_c.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mkexits.

   The user exit shared library will be built in the $IEFH/lib directory.

Note: If snplog() output is desired the exit must be compiled with –DDEBUGON or the "#define snplog(a,b,c)" statement within the exit must be commented out.

Related User Exits

TIRXINFO

UNIX and Linux Asynchronous Daemon User Exits

The following table summarizes the functions available through the user exits for the Asynchronous Daemon:

Asynchronous Daemon: Language: C		
User Exit Name	**Source Code**	**Description**
AEFSECEX	aefsecex.c	Security Validation executable invoked by the Asynchronous Daemon

Details for the preceding user exits follow in a separate section for each.

AEFSECEX—Asynchronous Daemon Security Exit

Source Code

aefsecex.c

Purpose

This exit is implemented as a stand-alone utility providing security validation support for the Transaction Enabler environment. The AEF Async Daemon passes the current flow's transaction code, User Id and Password to the aefsecex exit. The default implementation of this security process checks the user ID against the password file. This processing applies only to buffers not containing enhanced security data. For enhanced security CFBs it is assumed that the DPSs provide their own security using the server's Client Security Validation exit (TIRSECV) invoked by the server runtime. This security exit is enabled by providing the "-l" (lowercase L) option when starting the Asynchronous Daemon, AEFAD. AEFAD will execute the exit if enabled/required.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
transaction name	Input	The transaction code for the flow being processed.
user id	Input	User Id from the header area of the CFB.
password	Input	Password from the header area of the CFB.

Return Code

The following table gives a brief description of each of the return codes:

Return Code	Description
0	Security check succeeded
1	User ID not allowed
2	User ID was not found
3	Password is invalid
4	User ID is not authorized
5	Security system is unavailable
6	User ID has been suspended
7	User ID needs to reregister
8	Security timeout
9	Security internal error

Default Behavior

No security checking is performed; the exit is never executed.

Building on UNIX/Linux

The Asynchronous Daemon Security User Exit is built as a standalone executable, aefsecex. A prerequisite for building the executable, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run mksecex.

To utilize the aefsecex

The security exit is enabled by providing the '-l' (lowercase L) flag when starting the Asynchronous Daemon.

The aefsecex executable will be built in the $IEFH/bin directory.

Related User Exits

None

UNIX/Linux Server Middleware User Exits

WebSphere MQ Server Transport - UNIX and Linux User Exits

All supplied WebSphere MQ Server Transport user exits are written using the C programming language. The following table briefly describes the WebSphere MQ Exits:

WebSphere MQ Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_MQS_DPS_EXIT	cimqsvex.c	MQ Directory Services Exit
CI_MQS_DPC_SETREPORTOPTIONS	cimqsvex.c	Used to override report options set by the runtime.
CI_MQS_DYNAMICQNAME_EXIT	cimqsvex.c	Provide Queue Name that will be used when opening a dynamic queue.

Details for the preceding user exits follow in a separate section for each.

CI_MQS_DPC_SETREPORTOPTIONS – Override Put Queue Report Options Exit

```
void CI_MQS_DPC_setReportOptions(MQLONG *reportOptions);
```

Source Code

cimqsvex.c

Purpose

This exit can be used to override the set of report options defined for the WebSphere MQ Put Message Descriptor prior to the issuance of an MQPUT() operations.

The report options set by the runtime are described in the following table:

Report Option	Description
MQRO_EXCEPTION	This type of report can be generated when an exception occurs. For instance if a message is sent to another queue manager and the message cannot be delivered to the specified destination queue.
MQRO_EXPIRATION	An Expiration report. This type of report is generated by the queue manager if the message is discarded prior to delivery to an application because its expiry time has passed.
MQRO_PASS_MSG_ID	If a report is generated, the MsgId of the current message being processed is to be copied to the MsgId of the report message.
MQRO_COPY_MSG_ID_TO_CORREL_ID	Indicates the correlation ID of the report generated should equal the message ID of the request originally issued.
MQRO_DEAD_LETTER_Q	This option causes the original message to be placed on the dead-letter queue when an exception occurs.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*reportOptions	Input/Output	A pointer to a long value representing the currently defined report options to be used in the Put Message Descriptor.

Return Code

None

Default Behavior

The runtime specified report options are left unchanged.

Building on UNIX/Linux

The Websphere MQ DPS Override Put Queue Report Options User Exit is built as part of the shared library libmqssx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f smqsexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

CI_MQS_DPS_EXIT—MQSeries DPS Directory Services Exit

```
void CI_MQS_DPS_Exit (char   *qMgr,
    char   *srvQname,
    long   *numRequests,
    long   *getMsgTO,
    char   *serverName);
```

Source Code

cimqsvex.c

Purpose

The provided sample WebSphere MQ DPS Directory Services exit is an implementation of Transaction routing. Transaction routing is a conceptual process that lets cooperative flow data be routed from a Distributed Process Server (DPS) to a programmatically determined Distributed Process Server (DPS). This exit can use the input serverName, which is the server load module name calling this exit, to programmatically modify the output parameters.

The current cooperative request's local queue manager and put queue names can be overridden using this exit. Additionally a Get queue timeout value as well as a parameter specifying the servers multiple transaction behavior can be customized.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*qMgr	Input/Output	A character string containing the name of the local queue manager. By default, the application obtains this information from the model during generation. This exit can override this name.
*srvQName	Input/Output	A character string containing the name of the put queue connecting to the local queue manager. By default, the application obtains this information from the model during generation. This exit can override this name.
*numRequests	Input/Output	Specifies the number of transactions the server calling this exit can execute prior to the server shutting down. A mechanism to limit the number of transactions a server can execute. Default: -1, no limit to the number of transactions that can be serviced.
*getMsgTO	Input/Output	A long value representing the reply timeout applied to the get queue associated with the current flow request. Default: The default value is MQWI_UNLIMITED (-1), wait indefinitely. If set, the number represents milliseconds. This exit can override this name.
*serverName	Input	A character string containing the name of the server load module executing this exit.

Return Code

None

Default Behavior

The supplied sample does not implement dynamic transaction routing. For more information about default values for the various parameters, see the preceding Arguments section.

Building on UNIX/Linux

The Websphere MQ DPS Directory Services User Exit is built as part of the shared library libmqssx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f smqsexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

CI_MQS_DYNAMICQNAME_EXIT—Dynamic Queue Name Override Exit

```
void CI_MQS_DynamicQName_Exit (char   *dynamicQName);
```

Source Code

cimqsvex.c

Purpose

The Dynamic Queue Name exit allows override of the queue name that will be used when opening a dynamic queue. The resulting Dynamic Queue will obtain its attributes from the specified WebSphere MQ Model Queue name. The passed in dynamicQName area can be modified by placing a null terminated string of the value to be used as the dynamic queue name, including the use of valid WebSphere MQ pattern characters used to name dynamic queues.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*dynamicQName	Input/Output	A pointer to a character buffer, of length MQ_Q_NAME_LENGTH+1 (48 +1), that contains the default name of the dynamic queue as built by the WebSphere MQ runtime (that is, username.processid.threadid).

Return Code

None

Default Behavior

The dynamic queue name is not modified.

Building on UNIX/Linux

The Websphere MQ DPS Dynamic Queue Name Override User Exit is built as part of the shared library libmqssx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).

3. Run make /f smqsexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

Tuxedo - UNIX and Linux User Exits

All supplied Tuxedo Transport Server user exits are written using the C programming language. The following table briefly describes the Tuxedo Exits:

Tuxedo Transport: Language: C		
User Exit Name	**Source Code**	**Description**
CI_S_POST_END	Cistuxx.c	Gives You The Opportunity To Inspect Or Modify The View32 Buffer Representation Of The Export View Of The Target Procedure Step After The Transaction Ends, And Before It Is Sent Back To The Invoking Client.
CI_S_POST_SRVDONE	Cistuxx.c	Gives You The Opportunity To Perform Cleanup Actions After Server Shutdown.
CI_S_POST_SVRINIT	Cistuxx.c	Gives You The Opportunity To Perform Actions During Server Initialization (boot).
CI_S_POST-BEGIN	Cixtuxx.c	Gives You The Opportunity To Inspect Or Modify The Cooperative Flow Request Buffer After The Call To The Tuxedo Tpbegin Call.
CI_S_PRE_END	Cistuxx.c	Gives You The Opportunity To Inspect Or Modify The View32 Buffer Representation Of The Export View Of The Target Procedure Step Before The Transaction Ends, And Before It Is Sent Back To The Invoking Client.
CI_S_USER_DATA_IN	Cistuxx.c	Gives You The Opportunity To Inspect Or Modify The Cooperative Flow Request Buffer Prior To Passing It Onto The Target Tuxedo Service.
CI_S_USER_DATA_OUT	cistuxx.c	Gives you the opportunity to inspect or modify the cooperative flow return buffer after leaving the target Tuxedo service.

Details for the preceding user exits follow in a separate section for each.

CI_S_POST_END—Tuxedo After Transaction Termination Exit

```
void  ci_s_post_end( void * returnBuff,  void * globdata )
```

Source Code

cistuxx.c

Purpose

The ci_s_post_end user exit is invoked after a global transaction ends. The exit provides access to the View32 buffer representation of the export view of the target procedure step before it is sent back to the invoking client. The exit also provides access to the server runtime global data structure.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*returnBuff	Input /Output	A pointer to the View32 buffer that will be returned to the invoking client.
*globdata	Input/Output	A pointer to the server runtime global data structure.

Return Code

None

Default Behavior

The exit does not modify any data.

Building on UNIX/Linux

The Tuxedo After Transaction Termination User Exit is built as part of the shared library libtxsx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f stuxexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_S_PRE_END

CI_S_POST_SVRDONE—Tuxedo After Server Shutdown Exit

```
void  ci_s_post_srvdone( )
```

Source Code

cistuxx.c

Purpose

The ci_s_post_svrdone is invoked when the server application shuts down. Any clean up procedures can be performed in this function.

Arguments

None

Return Code

None

Default Behavior

The exit does not modify any data.

Building on UNIX/Linux

The Tuxedo After Server Shutdown User Exit is built as part of the shared library libtxsx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f stuxexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_S_POST_SRVINIT

CI_S_POST_SVRINIT—Tuxedo After Server Initialization Exit

```
void  ci_s_post_srvinit( int argc, char ** argv )
```

Source Code

cistuxx.c

Purpose

The ci_s_post_svrinit user exit is invoked by the server runtime during server initialization (boot). The server application command line argument and argument count are passed as parameters to the function. The function can be used to do whatever initialization process is required. If the return is -1, the initialization has failed, the server application discontinues its activity and quits immediately. If the function returns 0, the initialization is successful.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
argc	Input	Argument count
**argv	Input	A pointer to a set of command line arguments

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
zero (0)	Success
non-zero	failure

Default Behavior

The exit does not modify any data.

Building on UNIX/Linux

The Tuxedo After Server Initialization User Exit is built as part of the shared library libtxsx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f stuxexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_S_POST_SVRDONE

CI_S_POST_BEGIN—Tuxedo After Begin Transaction Exit

```
void  ci_s_post_begin( TPSVCINFO * svc, void * globdata )
```

Source Code

cistuxx.c

Purpose

The ci_s_post_begin user exit is invoked after a global transaction is initiated (after a call to the Tuxedo tpbegin API). This exit provides access to the TPSVCINFO structure. (For more information about TPSVCINFO, see the Tuxedo User Guide.) TPSVCINFO contains the View32 request buffer from the client as well as the designated service name for the target procedure step. The exit also provides access to the global data structure of the server runtime.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*svc	Input /Output	A pointer to TPSVCINFO structure.
*globdata	Input/Output	A pointer to the server runtime global data structure.

Return Code

None

Default Behavior

The exit does not modify any data.

Building on UNIX/Linux

The Tuxedo After Begin Transaction User Exit is built as part of the shared library libtxsx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f stuxexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_S_PRE_END

CI_S_POST_END

CI_S_PRE_END—Tuxedo Prior to Transaction Termination Exit

```
void  ci_s_pre_end( void * returnBuff, void * globdata )
```

Source Code

cistuxx.c

Purpose

The ci_s_pre_end user exit is invoked just prior to terminating a global transaction. This exit provides access to the View32 buffer representation of the export view of the target procedure step before it is sent back to the invoking client. This exit also provides access to the server runtime global data structure. The transaction context still exists at the point that this exit is invoked.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*returnBuff	Input /Output	A pointer to the View32 response buffer that is returned to the invoking client.
*globdata	Input/Output	A pointer to the server runtime global data structure.

Return Code

None

Default Behavior

The exit does not modify any data.

Building on UNIX/Linux

The Tuxedo Prior to Transaction Termination User Exit is built as part of the shared library libtxsx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f stuxexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_S_POST_END

CI_S_USER_DATA_IN—Tuxedo Inbound Flow Data Access Exit

```
void  ci_s_user_data_in( TPSVCINFO * svc, void * globdata )
```

Source Code

```
cistuxx.c
```

Purpose

The ci_s_user_data_in user exit is invoked by the server runtime upon service invocation and before the execution; control is passed to the dialog manager from the server runtime. The exit provides access to the TPSVCINFO structure. (For more information about TPSVCINFO, see the Tuxedo User Guide.) TPSVCINFO contains the View32 request buffer from the client and the designated service name for the target procedure step. The exit also provides access to the global data structure of the server runtime.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*svc	Input /Output	A pointer to TPSVCINFO structure.
*globdata	Input/Output	A pointer to the server runtime global data structure.

Return Code

None

Default Behavior

The exit does not modify any data.

Building on UNIX/Linux

The Tuxedo Inbound Flow Data Access User Exit is built as part of the shared library libtxsx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).

3. Run make /f stuxexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_S_USER_DATA_OUT

CI_WS_DPC_URL_Exit — Web Services DPC URL User Exit

```
void CI_WS_DPC_URL_Exit(char *url,
size_t urlMaxLen,
char *modelName,
char *modelShortName,
char *trancode,
char *trancodeAlt,
char *procName,
char *procNameAlt,
char *nextLoc
```

Purpose

This exit will be called from the C Web Services CoopFlow prior to performing a Web Service connection. It gives the user an opportunity to modify the Web Service endpoint destination URL.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*url	Input/Output	Web Service end point URL
urlMaxLen	Input	Maximum length of Web Service end point URL
*modelName	Input	Name of the model containing the target Procedure Step

Name	I/O	Description
*modelShortName	Input	Short name of the model containing the target Procedure Step
*tranCode	Input	Transaction code of the target Procedure Step being processed
*tranCodeAlt	Input	Alternative name for the transaction code of the target Procedure Step being processed
*procName	Input	Name of the target Procedure Step to be called
*procNameAlt	Input	Alternative name of the target Procedure Step to be called
*nextLoc	Input	Next Location system attribute as set using CA Gen action diagram statements

Return Code

None

Default Behavior

If the URL value is not modified in this user exit, its value prior to calling this exit is used.

Building on UNIX/Linux

The Web Services DPC user exit is built as part of the shared library libwscx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have the correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f cwsexit.plat, where plat is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

CI_S_USER_DATA_OUT—Tuxedo Outbound Flow Data Access Exit

```
void  ci_s_user_data_out(void * returnBuff, void * globdata );
```

Source Code

cistuxx.c

Purpose

The ci_s_user_data_out user exit is invoked before execution of the called Pstep completes. The exit provides access to the View32 buffer representation of the export view of the target procedure step before it is sent back to the invoking client. The exit also provides access to the server runtime global data structure.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*returnBuff	Input/ Output	A pointer to the View32 buffer that will be returned to the invoking client.
*globdata	Input/Output	A pointer to the server runtime global data structure.

Return Code

None

Default Behavior

The exit does not modify any data.

Building on UNIX/Linux

The Tuxedo Outbound Flow Data Access User Exit is built as part of the shared library libtxsx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f stuxexit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

CI_S_USER_DATA_IN

Web Services - UNIX and Linux User Exits

All supplied Web Services Middleware user exits are written using the C programming language. The following table briefly describes the Web Services Exits:

Web Services Middleware: Language: C

User Exit Name	Source Code	Description
CI_WS_DPC_Exit	ciwsclx.c	Programmatic runtime override of parameters (base URL and context type) for Web Service destination.
CI_WS_DPC_URL_ Exit	ciwsclx.c	Programmatic runtime override of URL for Web Service destination

CI_WS_DPC_URL_Exit—Web Services DPC User Exit

```
void CI_WS_DPC_Exit(char *baseURL,
size_t baseURLMaxLen,
char *contextType,
size_t contextTypeMaxLen,
char *modelName,
char *modelShortName,
char *trancode,
char *trancodeAlt,
char *procName,
char *procNameAlt,
char *nextLoc)
```

Purpose

This exit will be called from the C Web Services CoopFlow prior to performing a Web Service connection. It gives the user an opportunity to modify the Web Service endpoint destination by overriding the base URL and the context type.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*baseURL	Input/Output	Scheme, Domain and Port of a Web Service end point URL
baseURLMaxLen	Input	Maximum length of baseURL
*contextType	Input/Output	Part of the path of a CA Gen Web Service end point URL
contextTypeMaxLen	Input	Maximum length of contextType
*modelName	Input	Name of the model containing the target Procedure Step
*modelShortName	Input	Short name of the model containing the target Procedure Step
*tranCode	Input	Transaction code of the target Procedure Step being processed
*tranCodeAlt	Input	Alternative name for the transaction code of the target Procedure Step being processed

Name	I/O	Description
*procName	Input	Name of the target Procedure Step to be called
*procNameAlt	Input	Alternative name of the target Procedure Step to be called
*nextLoc	Input	Next Location system attribute as set using CA Gen action diagram

Return Code

None

Default Behavior

If the base URL and the context Type variables expected by the sample implementation of this exit are not defined, the default values that are defined during the packaging of the cooperative model or overrides from commcfg.ini are used.

Building on UNIX/Linux

The Web Services DPC user exit is built as part of the shared library libwscx.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have the correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f cwsexit.plat, where plat is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None

UNIX and Linux C Proxy User Exits

The following table summarizes the functions available through the user exits for C Proxy applications:

C Proxy: Language: C		
User Exit Name	**Source Code**	**Description**
WRSECDECRYPT	proxyxit.c	Client/Server Decryption Exit
WRSECENCRYPT	proxyxit.c	Client/Server Encryption Exit
WRSECTOKEN	proxyxit.c	Client Security Token Exit

Details for the preceding user exits follow in a separate section for each.

WRSECTOKEN—Client Security Token Exit

```
int WRSECTOKEN (char *clientUserid,
    char *clientPassword,
    char *trancode,
    char *nextLocation,
    BOOL *bClntMgrSecurity,
    long *tokenLen,
    char *token,
    char *failureMsg)
```

Source Code

proxyxit.c

Purpose

The Client Side Security Exit is invoked by the proxy runtime to let a user influence how client security data is processed by the proxy runtime code involved in servicing a cooperative flow. Specifically, this exit influences if the Common Format Buffer (CFB) request will contain a security offset and if that data populated in the security offset should be used by other runtime components such as the Client Manager or Communications Bridge when servicing the cooperative flow request.

The trancode and nextLocation variables are provided as input. These input values can be used by the user exit code to determine what return code value should be specified.

In addition to the return code value, this exit has the option of returning some fields as output data to the calling runtime code.

Note: For more information about the input and output fields of this exit routine, see Arguments. For a description on what the invoking proxy runtime will do because of receiving one of the expected return values, see Return Codes.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*clientUserid	Input/Output	A pointer to a character array that contains the value of the proxy ClientUserid attribute. This user exit can set this value by modifying the data area pointed to by this argument. The value assigned by this user exit cannot exceed 64 bytes.
*clientPassword	Input/Output	A pointer to a character array that contains the value of the proxy ClientPassword attribute. This user exit can set this value by modifying the data area pointed to by this argument. The value assigned by this user exit cannot exceed 64 bytes.
*trancode	Input	A pointer to a character array that contains the trancode associated with the flow being processed by the proxy runtime synchronous or asynchronous cooperative flow operation.
*nextLocation	Input	A pointer to a character array that contains the Next Location variable associated with the flow being processed by the proxy runtime synchronous or asynchronous cooperative flow operation.
*bClntMgrSecurity	Output	A pointer to an integer Boolean field that can be set to either TRUE or FALSE. The value of this field only has meaning if this user exit returns SecurityUsedEnhanced. TRUE indicates that the security data (Client User ID and Client Password) that is added to the security offset of the associated CFB should be used as the source of the UserID and Password by the Client Manager or Communications Bridge.

Name	I/O	Description
*tokenLen	Input/Output	On input, tokenLen is a pointer to a long integer field that contains the maximum length of the allocated token character buffer. The maximum token length is dependent on the available space remaining during the construction of the CFB. On return from the exit, the long integer pointed to by tokenLen should contain the actual length of data returned in the character array, which is pointed to by the token argument. **Note:** The use of a token is optional, and therefore, setting the long integer pointed to by tokenLen to zero indicates that a token is not specified by the user exit. The length value returned by this field only has meaning if this user exit returns SecurityUsedEnhanced.
*token	Input/Output	On input, token is a pointer to a character array that will accept a user specified security token. The use of a user specified security token is optional. The token data that is provided by this user exit will be provided to the server side TIRSECV security user exit. The security token returned by this field only has meaning if this user exit returns SecurityUsedEnhanced.
*failureMsg	Input/Output	The pointer to an 80-character array that can receive a user provided null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is displayed by the proxy runtime.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
SecurityNotUsed	Indicates to the runtime that the CLIENT_USER_ID, CLIENT_PASSWORD, and security token will NOT be used to populate any part of the cooperative flow request. The client side security variables will not be added to the CFB by the proxy runtime.

Return Code	Description
SecurityUsedStandard	Indicates to the runtime that at most eight (8) bytes of the CLIENT_USER_ID and at most eight (8) bytes of the CLIENT_PASSWORD data will be set into the CFB header. The associated request buffer will not contain a CFB Security Offset area, and will therefore, not contain a security token. Additionally, by not making use of the CFB Security Offset area, the Client User ID and Client Password values are not eligible for being encrypted.
SecurityUsedEnhanced	Indicates to the runtime that the CLIENT_USER_ID, CLIENT_PASSWORD, and the optional Security Token should be added to the CFB by way of the CFB Security Offset. Additionally, at most (8) bytes of the Client User ID value will be set into the CFB header.
SecurityError	Indicates to the runtime that an error was encountered by the user exit and that the processing of the associated request has failed. The error indication and message string returned using the failureMsg argument would be returned to the proxy runtime. The proxy runtime will popup an error message display indicating the failed request.

Default Behavior

The WRSECTOKEN user exit, as delivered with CA Gen, will return SecurityNotUsed. In addition, although not necessary, the user exit will set the long integer pointed to by the tokenLen pointer to zero, and set the Boolean field pointed to by the bClntMgrSecurity pointer to False.

Building on UNIX/Linux

The C Proxy Security Token User Exit is built as part of the shared library libprex.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).

3. Run make /f proxyxit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

The following are related user exits:

- TIRSECV
- WRSECENCRYPT
- WRSECDECRYPT

WRSECENCRYPT—Client Side Encryption Exit

```
int WRSECENCRYPT (char *trancode,
    char *nextLocation,
    char *clientUserid,
    long maxViewLen,
    long *encryptViewLen,
    unsigned char *encryptView,
    char *failureMsg)
```

Source Code

proxyxit.c

Purpose

The Client Side Encryption exit is called by the proxy runtime to provide the opportunity to encrypt a cooperative flow request from C Proxy applications. The data in the Common Format Buffer (CFB) that is eligible to be encrypted include the cooperative flow's view data and optional security offset area.

The user provides an encryption algorithm that consists of manipulating the data pointed to by encryptView. The encryptViewLen, on input contains the number of bytes eligible for being encrypted. The process of encryption cannot result in an encrypted buffer area that exceeds maxViewLen. If encryption is performed by this exit, EncryptViewLen must be updated with the length of the encrypted result. Additionally, this exit must return the EncryptionUsed return code value.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*trancode	Input	A pointer to a character array that contains the trancode associated with the synchronous or asynchronous cooperative flow being processed by the proxy runtime.
*nextLocation	Input	A pointer to a character array that contains the Next Location associated with the synchronous or asynchronous cooperative flow being processed by the proxy runtime.
*clientUserid	Input	A pointer to a character array that contains the value of the of the CLIENT_USER_ID variable associated with the flow being processed by the proxy runtime synchronous or asynchronous cooperative flow processing. The CLIENT_USESRID variable is optionally set by Action Language coded within the C Proxy code.
MaxViewLen	Input	A long field that contains the maximum available buffer space (in bytes) that the encrypted data can occupy.
*encryptViewLen	Input/Output	A pointer to a long field. On input, EncryptViewLen is the length of the current buffer space (in bytes) of the data eligible for being encrypted. On output, EncryptViewLen should be updated to contain the length of the encrypted data. The length of the encrypted result cannot exceed maxViewLen.
*encryptView	Input/Output	A character pointer to the starting location of the data eligible for being encrypted. The encrypted data must be copied to this same memory location.
*failureMsg	Input/Output	The pointer to an 80-character array that can receive a user provided null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is displayed by the proxy runtime.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
EncryptionNotUsed	Indicates to the runtime that the user exit did not perform any encryption to the provided data buffer.

Return Code	Description
EncryptionUsed	Indicates to the runtime that the user exit did perform encryption on the provided data. The runtime marks the CFB as being encrypted. An encrypted CFB will trigger the decryption counterpart user exit to be invoked by the target server manager. The server side decryption user exit is TIRDCRYP.
EncryptionFailure	Indicates to the runtime that an error was encountered by the user exit and that the processing of the associated request has failed. The error indication and message string returned using the failureMsg argument would be returned to the proxy runtime. The proxy runtime will pop up an error message display indicating the failed request.

Default Behavior

The WRSECENCRYPT user exit, as delivered with CA Gen, will return EncryptionNotUsed.

Building on UNIX/Linux

The C Proxy Encryption User Exit is built as part of the shared library libprex.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f proxyxit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

The following are related user exits:

- TIRNCRYP
- WRSECDECRYPT

WRSECDECRYPT—Client Decryption Exit

```
int WRSECDECRYPT (long maxViewLen,
    long *encryptViewLen,
    unsigned char *encryptView,
    char *failureMsg)
```

Source Code

proxyxit.c

Purpose

The Client Side Decryption exit is called by the proxy runtime when an encrypted response buffer is received from a target server.

The user provides a decryption algorithm that manipulates the data pointed to by encryptView. The encryptViewLen, on input contains the number of bytes available into which the encrypted buffer area can be decrypted. The process of decryption cannot result in a decrypted buffer area that exceeds maxViewLen. If decryption is performed by this exit, EncryptViewLen must be updated with the length of the decrypted result.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
maxViewLen	Input	A long field that contains the maximum available buffer space (in bytes) that the decrypted data can occupy.
*encryptViewLen	Input/Output	On input, EncryptViewLen is the current buffer space (in bytes) of the encrypted data. On output, EncryptViewLen should be updated to contain the length of the decrypted data. The length of the decrypted result cannot exceed maxViewLen.
*encryptView	Input/Output	A pointer to the starting location of the data eligible for being decrypted. The decrypted data must be copied back into this same memory location.

Name	I/O	Description
*failureMsg	Input/Output	The pointer to an 80-character array that can receive a user provided null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is displayed by the proxy runtime.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
DecryptionNotUsed	Indicates to the runtime that the user exit did not perform any decryption of the encrypted data buffer.
DecryptionUsed	Indicates to the runtime that the user exit successfully performed the decryption of the provided encrypted data.
DecryptionFailure	Indicates to the runtime that an error was encountered by the user exit and that the decryption processing has failed The error indication and message string returned using the failureMsg argument will be returned to the proxy Runtime. The proxy runtime will pop up an error message display indicating the failed request.

Default Behavior

The WRSECDECRYPT user exit, as delivered with CA Gen, will return DecryptionNotUsed.

Building on UNIX/Linux

The C Proxy Decryption User Exit is built as part of the shared library libprex.xx.*, where * is the shared library suffix depending on the UNIX system. As a prerequisite for building the shared library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).

3. Run make /f proxyxit.*plat*, where *plat* is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

The following are related user exits:

- TIRDCRYP
- WRSECENCRYPT

Chapter 4: z/OS User Exits

Changes to User Exits

In CA Gen Release 8.5, each user exit is its own stand-alone DLL. To eliminate the need to regenerate or re-link applications built with CA Gen Release 8.5 and to enable those applications to use the DLL user exits, the code in the previous release user exits invokes the new DLLs that contain the user exit code.

Note: Although the need to regenerate or relink applications built with CA Gen Release 8.5 has been eliminated, the customized user exits for those applications will need to be included in the equivalent user exit in CA Gen Release 8.5 and built as a user exit DLL using member MKUEXITS in the CEHBSAMP dataset.

Note: IMS server applications that use trace facilities must be re-linked to use the new TIRIDTRZ user exit DLL, called TIRIDTRM in previous releases.

The following table lists the old name, the new name, and the DLL name for each user exit.

Old User Exit Name		New User Exit Name		DLL Name
Source Member	**Exit - PGM**	**Source Member**	**Exit - PGM**	
TIRALLOC	TIRALLOC	TIRALLOX	TIRALLOX	TIRALLOZ
TIRBRTRY	TIRBRTRY	TIRBRTRX	TIRBRTRX	TIRBRTRZ
TIRBURTL	TIRBURTL	TIRBURTX	TIRBURTX	TIRBURTZ
TIRCDPTS	TIRCDPTS	TIRCDPTX	TIRCDPTX	TIRCDPTZ
TIRCROUT	TIRPROUT	TIRCROUX	TIRPROUX	TIRCROUZ
TIRCRTRY	TIRCRTRY	TIRCRTRX	TIRCRTRX	TIRCRTRZ
TIRCSGN	TIRCSGN	TIRCSGNX	TIRCSGNX	TIRCSGNZ
TIRCSYS	TIRSYSID	TIRCSYSX	TIRSYSIX	TIRCSYSZ
TIRCURTL	TIRCURTL	TIRCURTX	TIRCURTX	TIRCURTZ
TIRCUSR	TIRUSRID	TIRCUSRX	TIRUSRIX	TIRCUSRZ
TIRDATX	TIRDATX	TIRDATX	TIRDATX	TIRDATXZ
TIRDCRYP	TIRDCRYP	TIRDCRYX	TIRDCRYX	TIRDCRYZ
TIRDEVC	TIRDEVC	TIRDEVC	TIRDEVC	TIRDEVCZ

Old User Exit Name		New User Exit Name		DLL Name
Source Member	**Exit - PGM**	**Source Member**	**Exit - PGM**	
TIRDEVI	TIRDEVI	TIRDEVI	TIRDEVI	TIRDEVIZ
TIRDEVT	TIRDEVT	TIRDEVT	TIRDEVT	TIRDEVTZ
TIRDLCT	TIRDLCT	TIRDLCTX	TIRDLCTX	TIRDLCTZ
TIRELOG	TIRELOG	TIRELOGX	TIRELOGX	TIRELOGZ
TIRHELP	TIRHELP	TIRHELPX	TIRHELPX	TIRHELPZ
TIRIDTRM	TIRIDTRM	TIRIDTRX	TIRIDTRX	TIRIDTRZ
TIRIEX	TIRIEX	TIRIEX	TIRIEX	TIRIEXZ
TIRIROUT	TIRPROUT	TIRIROUX	TIRPROUX	TIRIROUZ
TIRIRTRY	TIRIRTRY	TIRIRTRX	TIRIRTRX	TIRIRTRZ
TIRISYS	TIRSYSID	TIRISYSX	TIRSYSIX	TIRISYSZ
TIRIURTL	TIRIURTL	TIRIURTX	TIRIURTX	TIRIURTZ
TIRIUSR	TIRUSRID	TIRIUSRX	TIRUSRIX	TIRIUSRZ
TIRMQTDX	TIRMQTDX	TIRMQTDX	TIRMQTDX	TIRMQTDZ
TIRMTQB	TIRMTQB	TIRMTQB2	TIRMTQB	TIRMTQBZ
TIRNCRYP	TIRNCRYP	TIRNCRYX	TIRNCRYX	TIRNCRYZ
TIRPTOKN	TIRPTOKN	TIRPTOKX	TIRPTOKX	TIRPTOKZ
TIRQCNTL	TIRQCNTL	TIRQCNTX	TIRQCNTX	TIRQCNTZ
TIRRETCD	TIRRETCD	TIRRETCX	TIRRETCX	TIRRETCZ
TIRSECR	TIRSECR	TIRSECRX	TIRSECRX	TIRSECRZ
TIRSECV	TIRSECV	TIRSECVX	TIRSECVX	TIRSECVZ
TIRSIPEX	TIRSIPEX	TIRSIPEX	TIRSIPEX	TIRSIPEZ
TIRSRTRY	TIRSRTRY	TIRSRTRX	TIRSRTRX	TIRSRTRZ
TIRSURTL	TIRSURTL	TIRSURTX	TIRSURTX	TIRSURTZ
TIRTERMB	TIRTERMB	TIRTERBX	TIRTERBX	TIRTERBZ
TIRTERMA	TIRTERMA	TIRTERMA	TIRTERMA	TIRTERAZ
TIRTERMB	TIRTERMB	TIRTERBX	TIRTERBX	TIRTERBZ
TIRTIAR	TIRTIAR	TIRCTIAX	TIRTIARX	TIRCTIAZ
TIRTIAR	TIRTIAR	TIRITIAX	TIRTIARX	TIRITIAZ
TIRTSYS	TIRSYSID	TIRTSYSX	TIRSYSIX	TIRTSYSZ

Old User Exit Name		New User Exit Name		DLL Name
Source Member	**Exit - PGM**	**Source Member**	**Exit - PGM**	
TIRUPPR	TIRUPPR	TIRUPPRX	TIRUPPRX	TIRUPPRZ
TIRXINFO	TIRXINFO	TIRXINFO	TIRXINFO	TIRXINFZ
TIRYYX	TIRYYX	TIRYYX	TIRYYX	TIRYYXZ

z/OS Blockmode User Exits—CICS

These user exits are used by CA Gen Blockmode, that is 3270, generated applications. Some of these exits are used by applications targeting CICS only, IMS only or TSO only, while for others the same exit is used by applications in more than one target. This information is indicated in the exit itself.

TIRCDPTX—Dynamic Plan TSQ Processing Exit

z/OS Dialog Managers use the CA Gen Dynamic Plan TSQ Processing Exit.

Source Code

This exit is used by CICS applications only. The source code for this exit is in CA Gen CEHBSAMP library, in member TIRCDPTX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRCDPTX is as follows:

```
01  RUNTIME-PARM1          PIC X.
01  RUNTIME-PARM2          PIC X.
01  Q-NAME                 PIC X(8).
01  ACTION-CODE            PIC 9.
01  GLOBDATA                    size 3645 bytes.
```

Purpose

This exit is called when the delete of the TSQ used by the Dynamic Plan Exit (TIRC$EXT) fails because the TSQ does not exist. It is used to return a flag to control how the runtime handles the missing TSQ condition.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if Translated.
Q-NAME	input	The name of the TSQ which we expected to find, but is missing.
ACTION-CODE	input/output	A 1 byte numeric field indicating how the runtime should handle the missing TSQ condition: 1 – Abend and rollback. 2 – Send an error message to CICS CSSL output and terminate normally, without a rollback. 3 – Send an error message to CICS CSSL output and terminate without an abend but with a rollback. 4 – Ignore the condition and terminate without an abend or rollback.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

The default processing of this exit is to return an ACTION-CODE of 2 which causes the runtime to handle the condition encountered by the CICS API command that deletes the TSQ, send a message to the CICS CSSL output and continue processing without rolling back any database changes done by the application.

Customizing the Exit

Copy the TIRCDPTX exit to one of your libraries and modify ACTION-CODE to return a value other than 2.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None

TIRCRTRX—Default Retry Limit Exit

z/OS Dialog Managers use the CA Gen Default Retry Limit Exit.

Source Code

This exit is used by CICS applications only. The source code for this exit is in CA Gen CEHBSAMP library, in member TIRCRTRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRCRTRX is as follows:

```
01  RUNTIME-PARM1              PIC X.
01  RUNTIME-PARM2              PIC X.
01  RETRY-TIMES                PIC S9(4) COMP.
01 GLOBDATA                    size 3645 bytes.
```

Purpose

This exit is called at the beginning of a CA Gen CICS blockmode application to enable the defined default value for the TRANSACTION RETRY LIMIT system attribute to be modified. The TRANSACTION RETRY LIMIT is initialized to this value at the beginning of each new transaction. This value may subsequently be modified by a SET TRANSACTION RETRY LIMIT statement in an action diagram.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if Translated.

Name	I/O	Description
RETRY-TIMES	input/ output	The maximum number of times the transaction execution is retried.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

If the Default Retry Limit Exit is not modified the TRANSACTION RETRY LIMIT is initialized to 10. If the Default Retry Limit Exit is used, it must not return a value greater than that specified in the Ultimate Retry Limit Exit.

Customizing the Exit

The TRANSACTION RETRY LIMIT is initialized to this value at the beginning of each new transaction. This value may subsequently be modified by a SET TRANSACTION RETRY LIMIT statement in an action diagram.

The TRANSACTION RETRY LIMIT is used to specify the maximum number of times a transaction is retried when one of the following events occurs:

- A RETRY TRANSACTION action diagram statement executes.
- A deadlock or timeout occurs trying to access a database, and no WHEN DATABASE DEADLOCK OR TIMEOUT statement was provided for that entity action statement.

In either of these cases, any uncommitted database updates are rolled back, and an attempt is then be made to execute the application again. Once the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, no more retries can occur, and the application will fails with a runtime error if the last retry attempt was unsuccessful.

Modify the copied exit as needed.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exit

TIRCURTX

TIRTIARX—DB2 Message Exit

z/OS Dialog Managers use the CA Gen DB2 Message Exit.

Source Code

The source code for the version of the exit used by CICS application is in CA Gen CEHBSAMP library, in member TIRCTIAX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRTIARX is as follows:

```
01  RUNTIME-PARM1                PIC X.
01  RUNTIME-PARM2                PIC X.
01  TIRFAIL-SQLCA                PIC X.
01  TIRTIAR-ERRORS               PIC X.
01  TIRTIAR-TEXT-LEN             PIC X.
01  GLOBDATA                     size 3645 bytes.
```

Purpose

The DB2 Message Exit is used by all applications targeting DB2 database on z/OS. The TIRFAIL subroutine of the Dialog Server calls the DB2 Message exit, TIRTIARX, whenever an unrecoverable DB2 failure occurs. TIRTIARX then calls the subroutine DSNTIAR to convert the SQL code into text. The messages returned by DSNTIAR are then merged with the runtime error messages.

TIRTIARX exit must be a DLL in order to be invoked by Gen applications, even by those using Compatibility option. DSNTIAR and DSNTIAC are provided by IBM as non-DLL modules. Therefore they need to be invoked by via TIRLGLOD.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.
TIRFAIL-SQLCA	input	SQLCA
TIRTIAR-ERRORS	input/output	Error message lines.
TIRTIAR-TEXT-LEN	input	Length of error message.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Behavior

As provided by CA Gen, the default exit dynamically calls DSNTIAR and is compatible with prior releases. However the sample code also contains examples of how to call DSNTIAR or DSNTIAC statically.

The call to TIRTIARX is made when TIRFAIL is building the table of messages and occurs prior to calling the default termination exit. For more information, see the Online Termination Exit and Batch Termination Exit.

Customizing the DB2 Message Exit

Copy the default exit to one of your own libraries. The member name is TIRCTIAX. The default exit includes example code for the four possible combinations of calls. There are dynamic and static calls of both DSNTIAR and of DSNTIAC. Simply comment out the default call and remove the comments from the one you want to use.

To statically call DSNTIAR or DSNTIAC, link the routine into the TIRCTIAZ DLL not the Gen application.

To dynamically call DSNTIAR or DSNTIAC build this routine as a non-DLL stand-alone executable and provide a CICS program definition (PPT) for it. This means that you need PPT definitions if you use the default TIRTIARX module. If TIRTIARX is customized to call DSNTIAC instead of DSNTIAR see IBM's CICS and/or DB2 documentation about using DSNTIAC.

When you have completed your modifications, install your exit.

Note: DSNTIAC is shipped as source code and must be assembled. If you intend to use it, see your DB2 or CICS systems programmer to ensure that is has been assembled and that a load module is available. If not, either the install of the application module will fail with an unresolved module at the time of the link if your call is static, or you will abend at runtime if your call is dynamic.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRCURTX—Ultimate Retry Limit Exit

z/OS Dialog Managers for CICS blockmode applications use the CA Gen Ultimate Retry Limit Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library, in member TIRCURTX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRCURTX is as follows:

```
01  RUNTIME-PARM1               PIC X.
01  RUNTIME-PARM2               PIC X.
01  ULTIMATE-RETRY-LIMIT        PIC S9(9)   COMP.
01  GLOBDATA                    size 3645 bytes.
```

Purpose

The Ultimate Retry Limit Exit is used by all applications targeting DB2 database on z/OS. The Ultimate Retry Limit Exit allows the user to specify a maximum value for the TRANSACTION RETRY LIMIT system attribute. This value may never be exceeded, either by a SET TRANSACTION RETRY LIMIT statement in an action diagram, or by the Default Retry Limit Exit.

For an explanation of when and how the TRANSACTION RETRY LIMIT system attribute is used see Default Retry Limit Exit.

This exit provides a safeguard in case the system attribute TRANSACTION RETRY LIMIT is set to an excessive value by an action diagram. Once the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, no more retries can occur, and the application will fail with a runtime error if the last retry attempt was unsuccessful.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.

Name	I/O	Description
ULTIMATE-RETRY-LIMIT	input/output	The absolute limit which is defaulted to 99.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

If the Ultimate Retry Limit Exit is not modified, the maximum value of TRANSACTION RETRY LIMIT will be 99. The Ultimate Retry Limit Exit may be modified to return a value of zero to suppress all retry attempts.

Customizing the Exit

Copy the TIRCURTX exit to one of your libraries and modify the Ultimate-Retry-Limit to the appropriate value.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRSYSIX—System ID Exit

z/OS Dialog Managers use the CA Gen System Identification Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library in member TIRCSYSX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRSYSIX is as follows:

```
01  LOCAL-SYSTEM-ID             PIC X(8).

01  GLOBDATA                    size 3645 bytes
```

This exit contains CICS API calls, which require it to be processed by the Translator. The Translator automatically includes data structures for DFHEIBLK and DFHCOMMAREA in the place of RUNTIME-PARM1 and RUNTIME-PARM2 thus RUNTIME-PARMx are not specified.

Purpose

This exit is called by all CICS applications.The purpose of TIRSYSIX is to enable logic that lets the same application be implemented on multiple systems and perform processing specific to each system targeted.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
DFHEIBLK	input	Automatically included by the Translator.
DFHCOMMAREA	input	Automatically included by the Translator.
LOCAL-SYSTEM-ID	output	The identifier of the system where the application is executing.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

The default processing of the exit is to issue the CICS Assign Sysid command to retrieve the system ID. If successful, the retrieved ID is returned; otherwise, the literal CICS is returned.

Customizing the Exit

Copy the TIRCSYSX member to one of your libraries and modify to populate the LOCAL-SYSTEM-ID as required by the application.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRALLOX
- TIRPROUX

TIRUSRIX—User ID Exit

z/OS Dialog Managers use the CA Gen User Identification Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library in member TIRCUSRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRUSRIX is as follows:

```
01  FILLER-PARM                      PIC X.
01  TIRUSRID-PARM.
    05  IET-USER-ID                   PIC X(8).
    05  IET-USER-ID2                  PIC X(8).
01 GLOBDATA                          size 3645 bytes.
```

This exit contains CICS API calls, which require it to be processed by the Translator. The Translator automatically includes data structures for DFHEIBLK and DFHCOMMAREA in the place of RUNTIME-PARM1 and RUNTIME-PARM2 thus RUNTIME-PARMx are not specified.

Purpose

This exit is called by all CICS applications.The purpose of TIRUSRIX is to obtain the userid and terminal ID of the executing application so that these values can be used as part of the key for the DB2 Profile Table and in the application itself.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description		
DFHEIBLK	input	Automatically included by the Translator.		
DFHCOMMAREA	input	Automatically included by the Translator.		
FILLER-PARM	input	Not used.		
TIRUSRID-PARM	output	A pointer to a structure containing the following items:		
		IET-USER-ID	output	The userid to be used in the application.
		IET-USER-ID2	output	The ID to be used as part of the Profile Table key.

Return Code

No explicit return code value is set by the user exit.

Default Processing

There are two possible implementations for this exit.

The default processing of the exit is coded for applications that execute with a terminal facility, these are blockmode and servers that use SNA and ECI. In this case, the exit checks that the terminal ID and user ID values are present and these values are returned. If only the terminal ID is present, its value is returned as both terminal ID and user ID. If there is no terminal ID value, the CICS Task ID is returned as both terminal ID and user ID.

The exit also contains sample code that can be used for applications that execute without a terminal facility, these are servers that use TCP/IP and MQSeries. For these applications, if the user ID and terminal ID are present the exit returns these values. If only the user ID is present, it is returned and the CICS Task ID is returned for the terminal ID. If only the terminal ID is present, its value is returned for both fields. If neither user ID nor terminal ID is present, the CICS Task ID is returned for both.

Customizing the Exit

Copy the TIRUSRIX to one of your libraries and modify to populate either IET-USER-ID or IET-USER-ID2 as required by the application.

Note: IET-USER-ID is used by the application as its User Identifier while IET-USER-ID2 is used as part of the Key to the RPROF (Profile Manager) Table.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

TIRSECRX

TIRSECRX—Security Interface Exit

z/OS Dialog Managers use the CA Gen Security Interface Exit.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRSECRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                          PIC X.
01  RUNTIME-PARM2                          PIC X.
01  TIRSECR-CMCB.
    03  TIRSECR-USERID                     PIC X(8).
    03  TIRSECR-TRANCODE                   PIC X(8).
    03  TIRSECR-TERMINAL-ID                PIC X(8).
    03  TIRSECR-SYSTEM-ID                  PIC X(8).
    03  TIRSECR-LOAD-MODULE                PIC X(8).
    03  TIRSECR-PSTEP-NAME                 PIC X(32).
    03  TIRSECR-DIALECT                    PIC X(32).
    03  TIRSECR-RETURN-CODE                PIC XX.
    03  TIRSECR-FAILURE-MSG                PIC X(80).
01  GLOBDATA                               size 3645 bytes.
```

Purpose

This exit is used by all CICS applications. The purpose of the TIRSECRX exit is to allow transaction-level security checking to be implemented. The Dialog Manager calls the Security Interface Exit when a transaction is started and before execution of a dialog flow. This allows transaction-level security checking to be implemented. After it has been enabled, the Dialog Manager executes the security interface exit automatically, at the relevant points, without any intervention by a programmer, when invoking any load modules in a business system.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description	
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.	
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.	
TIRSECR-CMCB	input/ output	A structure containing the following items:	
	TIRSECR-USERID	input	The userid under which this transaction is executing, as provided by the TIRUSRIX exit.
	TIRSECR-TRANCODE	input	The load module transaction code.
	TIRSECR-TERMINAL-ID	input	The terminal ID used by this transaction, spaces if this is a non-terminal transaction.
	TIRSECR-SYSTEM-ID	input	The system ID where this transaction is executing, as provided by the TIRSYSIX exit.

Name	I/O	Description
	TIRSECR-LOAD-MODULE	input	The load module name.
	TIRSECR-PSTEP-NAME	input	The Procedure Step name.
	TIRSECR-DIALECT	input	The dialect used by this application.
	TIRSECR-RETURN-CODE	output	A 2-byte character field returning the result of the security check. The following values are supported: SPACES—TIRSECR-ALL-OK Anything else—failure
	TIRSECR-FAILURE-MSG	output	An 80-byte character field, to be populated by this exit, to describe the failure with a message of choice.

Return Code

Update TIRSECR-RETURN-CODE with the relevant value.

Default Processing

The default processing of this exit is to do no security checking and to return TIRSECR-ALL-OK as the return code.

Customizing the Exit

Copy the TIRSECRX exit to one of your libraries and modify to perform security checking as required by the application. Ensure that TIRSECR-RETURN-CODE is set to spaces when the security check is successful or some other value to indicate failure. If a message describing the violation is returned in TIRSECR-FAILURE-MSG, the Dialog Manager will pass it to TIRTERMA.

Building on z/OS

For information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRUSRIX
- TIRSECVX
- TIRELOGX
- TIRTERMA

TIRQCNTX—TSQ Profile Manager Exit

z/OS Dialog Managers use the CA Gen TSQ Profile Manager Exit.

Source Code

The source code for this exit is in the Gen CEHBSAMP library in member TIRQCNTX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The parameter list used by TIRQCNTX is as follows

```
01 RUNTIME-PARM1                    PIC X.
01 RUNTIME-PARM2                    PIC X.
01  TIRQCNTL-CMCB.
    05  TIRQCNTL-QUEUE-NAME         PIC X(8).
    05  TIRQCNTL-STORAGE-TYPE       PIC X.
01 GLOBDATA                         size 3645 bytes.
```

Purpose

This exit is used by CICS applications only. The purpose of the TIRQCNTX is to allow the user to override the name of the queue used for the temporary storage queue profile and the type of storage used for the queue.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.
TIRQCNTL-QUEUE-NAME	input/ output	Name of the temporary storage queue used for the profile manager.
TIRQCNTL-STORAGE-TYPE	input/ output	Type of storage where the queue will reside.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

The default action for the exit is to leave the TSQ Profile Manager queue name , that consists of an internal application ID and the LTERM ID, unchanged and set the default storage type to MAIN.

Customizing the Exit

Copy the TIRQCNTX exit to one of your libraries and modify as required. The exit does not use CICS commands so it does not need to be translated for CICS and it specifies the RUNTIME-PARM1 and RUNTIME-PARM2 in both the Linkage Section and the Procedure Division statement. Ensure these are removed if CICS API calls are added.

The TIRQCNTL-QUEUE-NAME needs to be unique per CICS region.

The TIRQCNTL-STORAGE-TYPE can be set to either USE-AUXILIARY-STORAGE instead of USE-MAIN-STORAGE.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRDATX—Date and Time Services Exit

z/OS Dialog Managers use the CA Gen Date and Time Services Exit.The Date and Time Services Exit can be used to intercept, adjust, or validate system dates and times. This exit is provided to allow user modification and customization of date and time processing.

Source Code

The source code for this exit is in the Gen CEHBSAMP library in member TIRDATX. The sample exit provided is written in Assembler and uses standard OS Linkage.

The parameter list used by TIRDATX is as follows:

```
PARMRT1  DS    A
PARMRT2  DS    A
PARMCMCB DS    A
PARMWORK DS    A
PARMGDTA DS    A
```

Purpose

This exit is only used by Gen applications that use Standard Mapping facilities, not by Enhanced Map.This exit receives control for some but not all date and time services. Only services that acquire, or manipulate the date and time, where that date or time was acquired from the system, or where validation is involved, invoke this exit.

This exit is not invoked for the following conditions:

- Services involving conversion from one form to another does not invoke this exit.
- If some error condition exists. For example, if the clock is not set, the date and time services return the error directly to the requester and do not call this exit.

- For validation, if the value is not valid, the failure is returned to the requester and the exit is not called.
- If this exit changes a date or time and requests re-validation, and the value is in error, the error is returned to the requester and the exit is not called.

Note: If the date and time is modified by the exit, the exit must indicate this by returning the appropriate return code. Return codes that are invalid (not one of the listed values) will be ignored and the processing is as if the exit returned zero (0). Therefore it is imperative that you not take advantage of any behavioral aspects not explicitly documented here or in the sample code since future releases could change the operation.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
PARMRT1	input	This is DFHEIBLK automatically included if translated.
PARMRT2	input	This is DFHCOMMAREA automatically included if translated.
PARMCMCB	input/ output	Address of the Date Communication Control Block
PARMWORK	input/ output	Address of a 256 byte workarea
PARMGDTA	input	Address of GLOBDATA

Return Code

Only some of the date and time services functions are available to this exit, these are indicated by the value of the DREQ field. Before returning this exit must restore registers 0-14 to their values on entry and update register 15 with the return code that controls how the date and time services functions continue processing.The return code varies depending on the request, as follows:

DREQ - Service Request DREQVAL

- 0—Use the system date/time as stored in DCMCB. The exit has not modified these values and accepts them as they are.
- 4—The exit has modified the date/time stored in the DCMCB and requests that the validation be re-executed for these values.

  Note: The exit will be called again after validation is complete.

- 8—The exit has modified the date/time stored in the DCMCB and does not require the values be re-validated. The modified values are returned to the requester.
- 12—The exit requests that the date/time service fail the request. This is returned to the service requester as if the date and/or time were invalid.

DREQ - Service Request DREQSD, DREQAS, DREQVTS, and DREQST

- 0—Use the system date/time as stored in DCMCB.
- 4—The exit requests the date, time, or timestamp value be recomputed. If the exit has modified any of these values, the modifications are discarded and the values computed from the system clock. For DREQAS, the DINC value represents the number of days to be added to the date. The exit is called again after the date and time have been recomputed.
- 8—The exit has modified the date/time stored in the DCMCB and does not require the date/time services recompute the associated values. The modified and unmodified values are returned to the service requester unchanged.
- 12—The exit requests that the date/time service fail the request. This is the same as if the clock was not set or was damaged.

Default Processing

Sample Code

Sample code has been provided as commented out blocks of code. This is as an example only. To use the sample code, you must remove the comments. Sections that are specific to a particular system, such as CICS, are indicated by comments preceding and following the code. Common code that should be used by both examples is also indicated.

The sample code provides the ability to respond to request code 1 (get system date) and request code 7 (get current timestamp). The date information is read from TSQ or a file and is used to change the year, month, and day. The timestamp information (hours, minutes, seconds, microseconds) is read and left unchanged.

Delivery Configuration

As configured, the sample code will read the required date from a CICS temporary storage queue when generated for CICS. This exit can also obtain values from DB2 table lookups.

To use this facility, you must change the source code to set a local variable as appropriate. If CICS is True (CICS mode), process using the CICS Translator and assembler. If CICS is False, use the assembler but do not use the CICS Translator.

Registers

Register 14 contains the address that control is to be returned to, and Register 13 contains the address of a savearea set up for the exit's usage. All registers must be saved on entry. Register 15 must be updated with the return code and all other registers must be restored on return.

Customizing the Exit

You can customize the exit to perform your specific needs. The following paragraphs provide guidelines to be observed when modifying this exit. Be sure to read all notes provided with the sample code for the latest information on using this exit.

Testing the DREQ Field

The exit must test the DREQ field of the Date CMCB to determine the service request made of the Date/Time routine. This is used to customize the exit based on your needs. For example, if you wish to perform local validation of dates only, the request of interest is DREQVAL. For all other requests, the exit must return a zero.

Service Requests Intercepted by the Date and Time Services Exit

The service requests intercepted by this exit are:

- DREQVAL—Request date and/or time validation
- DREQSD—Return the current system date and time
- DREQAS—Add a specified increment to the date value
- DREQST—Return the current timestamp
- DREQVTS—Validate the timestamp provided
- DREQST—Return the current system timestamp

Modifying Date and Time

If the exit is used to modify date or time, the exit must modify the appropriate fields for the service request. Different service requests use different areas of the Date CMCB as their input, and place their output in various fields.

I/O Format

The format of input and output data are indicated in the CMCB fields DDATEF and DTIMEF. These values should be examined to determine the format of the data to be stored, or to be used as input by the exit.Fields in the Date CMCB

Other fields in the CMCB have various meanings and formats as described in the following paragraphs.

DDATE

This field contains the binary date value. It is treated as a signed decimal number and converted to binary. The format is specified by the field DDATEF and cannot be changed. This field can be in one of the following formats:

- YYYYMMDD—Four digit year, two digit month, and two digit day
- YYMMDD—Two digit year (the century is omitted), two digit month, and two digit day
- CYYMMDD—One digit century code, two digit year, two digit month, and two digit day

Note: The one digit century code (C) is a number from 0 to 9, inclusive. The century ranges that can be represented are from 1600 to 2599, inclusive. The century codes are:

0 = 19XX, 1 = 20XX, 2= 21XX,3 = 22XX, 4 = 23XX,

5 = 24XX, 6 = 25XX, 7 = 16XX, 8 = 17XX, 9 = 18XX.

DDATEF

This field contains an indicator of the format of the DDATE field’s content and cannot be changed.

DTIME

This field contains the binary time value. The time is treated as a signed decimal number with the format HHMMSSTH, or HHMMSST, or HHMMSS with the following conventions:

- HH—Hours
- MM—Minutes
- SS—Seconds

- T—Tenths of seconds
- H—Hundredths of seconds

The format used is specified by the DTIMEF field.

DTIMEF

This field contains an indicator of the format of the DTIME field's content and cannot be changed.

DTSTAMP

This field contains the zoned decimal time stamp value in a fixed format of YYYYMMDDHHMISSNNNNNN with the following conventions:

- YYYY—Four-digit year
- MM—Two-digit month
- DD—Two-digit day
- MI—Two-digit minutes
- SS—Two-digit seconds
- NNNNNN—Six-digit microseconds

DINC

This field contains a signed binary increment to be added to the date value in DDATE when DREQAS service is requested. It is unused in all other cases. A negative value will result in a date.

The following table describes service requests and the fields they use:

Service Request (DREQ)	Input	Output	Applicable Notes
DREQVAL	DDATE, DTIME		1, 4, 5
DREQSD		DDATE, DTIME	2, 5
DREQAS	DDATE, DTIME, DINC	DDATE, DTIME	2, 5
DREQVTS	DTSTAMP		4, 5
DREQST		DTSTAMP, DDATE, DTIME	2, 3, 5

Note:

1. Date and/or Time validation can be skipped if the appropriate field is set to zero. For example, if DDATE is zero, then the Date validation is skipped.
2. Initial processing obtains the current date and time using the system clock and adjusts the value based on the time zone adjustment. If the request is DREQAS, then the DINC value is added to the number of days prior to computing the Gregorian date, and then the DDATE / DTIME fields are computed. If the exit requests that the values be reprocessed, any modification that the exit made to the DDATE / DTIME fields is discarded and the values recomputed from the system clock. DINC can be altered if the request was DREQAS.
3. If the request is DREQST, then the system time stamp values are computed from the clock values.
4. Validation returns a code to the requester indicating the validity of the date/time/time stamp. If the value is valid, the exit is called or recalled if the exit requested the validation be reprocessed.
5. The formats of input and output data are indicated in the CMCB fields DDATEF and DTIMEF. These values should be examined to determine the format of the data to be stored, or to be used as input by the exit.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None

TIRDEVC—Device Characteristics Exit

z/OS Dialog Managers use the CA Gen Device Characteristics Exit.

Source Code

The source code for this exit is in the Gen CEHBSAMP library in member TIRDEVC. The sample exit provided is written in Assembler and uses standard OS Linkage.

The parameter list used by TIRDEVC is as follows:

```
PARMRT1  DS    A
PARMRT2  DS    A
EXTATTRA DS    A
DEVUSER  DS    A
DIALECT  DS    A
GLOBDATA DS    A
```

Purpose

This exit is only used by Gen applications that use Standard Mapping facilities, not by Enhanced Map. The Device Characteristics exit, TIRDEVC, is called every time a message is sent from or received by an application. This exit provides the runtime data stream processing routines for the definition of the specific device characteristics.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
DFHEIBLK	input	Automatically included by the Translator.
DFHCOMMAREA	input	Automatically included by the Translator.
Parm 3	input/ output	Device Characteristics
DEVCAP	input/ output	TMOHDLCT (dialect)/workarea
GDTA	input/ output	Address of GLOBDATA

TMOHDLCT is a pointer to the eight-character dialect name as returned from the User Dialect exit, TIRDLCTX. This value represents the current selected dialect. The default exit returns a default dialect value for this parameter.

The DEVCAP is a pointer to a 256-byte structure defined for the return of the device attributes.

Return Code

The Device Characteristic structure contains the returned device characteristics. The fields in this structure are as follows:

EXTPARM		Returned Device Capabilities
MAXROWR	DS H	(24/32/43/27) maximum number of screen rows
MAXCOLR	DS H	(80/132) maximum number of screen columns
EXTDSR	DS CL1	(0/255) 0= No Extended Data Stream support
EXTCLRR	DS CL1	(0/255) 0= Base Color, 255 = Extended Color
EXTHIGHR	DS CL1	(0/255) 0= No Highlight, 255 = Highlighting
EXTGRID	DS CL1	(0/255) 0= No Grid Line, 255 = Grid Line
EXTDBCS	DS CL1	(0/255) 0= No DBCS DISPLAY or ENTRY
EXTSCS8	DS CL1	(0/255) 0= No DBCS Set F8, SCS'8' for DBCS
XMIXENT	DS CL1	(0/255) 0= No Mixed (SBCS/DBCS) entry
XINEDIT	DS CL1	(0/255) 0= No INPUT EDITING ATTRIBUTE support
XOUTXLAT	DS A	Pointer to 256 byte Output Translate Table
XINPXLAT	DS A	Pointer to 256 byte Input Translate Table
	DS CL235	Filler MUST BE ZERO

Default Processing

The maximum row and column values are derived from the 3270 model type. At this time, CA Gen supports only IBM 3270 model 2 (24 x 80).

Extended Data Stream support and other extended attribute capabilities of the terminal are derived from query or any other user defined method of retrieving the terminal status. If Extended data stream is not enabled, then no extended data stream functions are built into the outbound data stream. If Double Byte Character Support (DBCS) is not enabled, then no DBCS data is placed in the outbound data stream. If MIXENT is not enabled, all mixed entry fields are built as Single Byte Character Support (SBCS) only fields in the data stream.

Additional information is available in the vendor documentation on National Language Support (NLS).

Translate Tables

The output (OUTXLAT) and input (INPXLAT) tables are standard 256 byte translate tables in a format suitable for the translate (TR) operations code (op code). OUTXLAT is used when the current device does not support the same code page as the application and encyclopedia. This means that a difference exists in the code points for the encyclopedia and application database and the code points for the device. The translate table needs to convert the code points in the output data stream to the correct code points for the current device to display the correct glyphs. INPXLAT is used when data is received from the terminal to convert the code points back to the appropriate values for the application database and encyclopedia.

If the device supports the same code page as the application and database, then OUTXLAT and INPXLAT should be set to ZERO (0) to suppress any code point conversion.

For example, if the current device is a Japanese 557x terminal supporting code page 930 (uppercase Roman only) and the application prompts contain lower case Roman letters, the translate tables must perform inbound and outbound translations.

Outbound, the translate table performs monocasing (from lowercase to uppercase), and translates the application database code points to the device code points. This displays the correct glyphs on the device.

Inbound, the translate table translates the device code points to the application database and encyclopedia code points for proper storage. This prevents corruption of the data in the database.

To accomplish the translation process in the preceding example, set OUTXLAT to point to a table that converts lowercase code points to uppercase. Set INPXLAT to a table that translates device Katakana back into the code point values needed in the application database.

Customizing the Exit

Copy the default exit from the CA Gen sample library to a separate library. The member name is TIRDEVC. You can customize this exit to accept input from the User Dialect exit (TIRDLCTX) to change the code page during production.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

TIRDLCTX

TIRDLCTX—User Dialect Exit

z/OS Dialog Managers use the CA Gen User Dialect Exit.

Source Code

The source code for this exit is in the CA Gen CEHBSAMP library member TIRDLCTX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                                PIC X.
01  RUNTIME-PARM2                                PIC X.
01  TIRDLCT-CMCB.
    03  TIRDLCT-USERID                          PIC X(8).
    03  TIRDLCT-TERMINAL-ID                     PIC X(8).
    03  TIRDLCT-SYSTEM-ID                       PIC X(8).
    03  TIRDLCT-RETURN-DIALECT                   PIC X(8).
01  GLOBDATA                                    size 3645 bytes.
```

Purpose

This exit is used by all applications. The purpose of the TIRDLCTX exit is to supply the current user's dialect to the application. It is meaningful for multilingual applications.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description	
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.	
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.	
TIRDLCT-CMCB	input/ output	A structure containing the following items:	
	TIRDLCT-USERID	input	The userid under which this transaction is executing, as provided by the TIRUSRIX exit.

Name	I/O	Description	
	TIRDLCT-TERMINAL-ID	input	The terminal ID used by this transaction, spaces if this is a non-terminal transaction.
	TIRDLCT-SYSTEM-ID	input	The system ID where this transaction is executing, as provided by the TIRSYSIX exit.
	TIRDLCT-RETURN-DIALECT	input	The dialect used by this application.

Return Code

No explicit return code is set by the user exit.

Default Processing

The default processing of this exit is to return a dialect name of DEFAULT.

Customizing the Exit

Copy the default exit from the CA Gen CEHBSAMP library to one of your libraries. The member name is TIRDLCTX. For multilingual support, modify this module to return the appropriate dialect for a user. The dialect returned is the one selected using the Dialect Definition option of the Design Toolset. If none is selected or returned, the default dialect is used.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

TIRDEVC

TIRUPPRX—Uppercase Translation Exit

z/OS Dialog Managers use the CA Gen Uppercase Translation Exit. This exit is also called the Lower-to-Uppercase Conversion Exit.

Source Code

This exit is used by single byte and double byte applications. When used by double byte applications an alternate entry point TIRUPDBx is used. The source code for this for this exit is in CA Gen CEHBSAMP library in member TIRUPPRX. The sample exit is written in COBOL and uses OS linkage.

The Parameter list used by TIRUPPRX is as follows:

```
01  RUNTIME-PARM1                    PIC X.
01  RUNTIME-PARM2                    PIC X.
01  XLATE-TABLE-NAME                PIC X(8).
01  XLATE-LEN                       PIC S9(4) COMP.
01  XLATE-DATA                      PIC X(4096).
01  GLOBDATA                         size 3645 bytes.
```

Purpose

The purpose of the Uppercase Translation User Exit is to translate character input from lowercase to uppercase. It contains a table of paired lower and uppercase characters. This exit is called by the Dialog Manager to translate the lower caser trancode to upper case, by the TIRFUPPR Function to translate the designated data to upper case and by the Standard Map runtime to translate the identified input data to upper case.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.
XLATE-TABLE-NAME	input	Name of the translation table to be used.
XLATE-LEN	input	Length of data to be translated.
XLATE-DATA	input/output	Data to be translated.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code value is defined for this exit.

Default Processing

The default processing of this exit is to convert lower case characters to upper case using a table named DEFAULT that contains the English character set(A-Z).

Customizing the Exit

Copy the default exit from the CA Gen CEHBSAMP library to one of your own libraries. The member name is TIRUPPRX.

The exit supports both single byte and double byte languages. Adding support for DBCS is done in the same way as for single byte.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRYYX—Two-Digit Year Input Edit Exit

z/OS Dialog Managers use CA Gen Two-Digit Year Input Exit.

Source Code

The source code for this exit is in the CA Gen CEHBSAMP library member TIRYYX. The sample exit provided is written in Assembler and uses standard OS Linkage.

The parameter list used by TIRYYX is as follows:

```
EXTCB     DS     A
WORKAREA  DS     A
GLOBDATA  DS     A
```

Purpose

This exit is used by CA Gen Standard Map applications only. The purpose of the TIRYYX exit is to process two-digit or YY-style date input and set the century part using any chosen algorithm to implement logic to handle the century part of the date.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
EXITCB	input	Address of the Date Communication Control Block
WORKAREA	input	Address of a 256 byte workarea
GLOBDATA	input	Address of GLOBDATA

Return Code

Before returning this exit must restore registers 0-14 to their values on entry and update register 15 with a value of 4 to indicate that the YY modified by the exit should be used. Any other value, including 0, indicates the original values passed to the exit are acceptable to continue processing.

Default Processing

The exit contains sample code for 2 algorithms but neither are executed. By default the exit returns a value of 0, indicating that no changes were done by the exit.

Customizing the Exit

Copy the TIRYYX exit to one of your libraries.

Internally, CA Gen handles four-digit year dates correctly assuming the user application uses a YYYY edit pattern throughout. If the user interface is designed to accept a two-digit date entry, and defaulting to the current century is not acceptable, use this exit to implement logic to get the required behavior for defaulting the century part of the date. The exit is called to process either a DATE or TIMESTAMP field which utilizes a 2-digit year (YY) in the edit pattern associated with the field. An indicator is set in the exit control block to indicate if the value is a date or timestamp.

Modify the exit to use one of the provided algorithms or add your own as required by your applications.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRTERMA—Termination Exit

The CA Gen Termination Exit is called by z/OS Dialog Managers when a fatal runtime error is encountered.

Source Code

TIRTERMA Termination Exit is used by all non-cooperative applications targeting z/OS. The source code is in CA Gen CEHBSAMP library, in member TIRTERMA. The sample exit provided is written in COBOL and uses standard OS Linkage.

The parameters passed between the fail routine and the termination exits are defined via structure TERM-EXIT-PARM-LIST. This structure is included via copy member CBLTERM, which is also in the CEHBSAMP library.

The Linkage Parameter list used by TIRTERMA is as follows:

```
01  RUNTIME-PARM1              PIC X.
01  RUNTIME-PARM2              PIC X.
01  TERM-EXIT-PARM-LIST        structure defined in CBLTERM.
01  GLOBDATA                   size 3645 bytes.
```

Purpose

The purpose of the TIRTERMA exit is to control how fatal runtime errors are handled by the Dialog Manager.

Runtime errors are either fatal or non-fatal errors. When a non-fatal error occurs, such as invalid user input, the Dialog Manager displays an error message on the transaction screen. You can correct the error and continue processing the transaction.

When a fatal error occurs, transaction processing is terminated. The Dialog Manager executes a fail routine that backs out changes by performing the necessary rollbacks of the databases. The fail routine then calls a termination exit that determines what diagnostic (error) information is displayed and where it is displayed.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.
TERM-EXIT-PARM-LIST	input/ output	Structure of parameters for termination and failure routine. The items in this structure are described in the CBLTERM Field Definitions. Description of these fields follow.
GLOBDATA	input	Global data, used internally.

The structure TERM-EXIT-PARM-LIST is defined via copy member CBLTERM. Two items in this structure control processing. These items are:TERM-STATUS-CODE

When TIRFAIL calls TIRTERMA, TERM-STATUS-CODE is used to control what TIRFAIL does next.

The following table provides a description of each TERM-STATUS-CODE value:

Value	Description
’ ’ (space) or 0 (zero)	TIRFAIL displays the message and redisplays the previous screen with TERM-DEFAULT-MSG in the error message field.
1	This value indicates that TIRTIRMA has handled the messages and will not display them. It will, however, redisplay the previous screen with TERM-DEFAULT-MSG in the error message field.
2	Thisvalue indicates that TIRTIRMA has handled everything. TIRFAIL does not display the messages and does not redisplay the previous screen.

TERM-FAIL-TYPE

The following table contains a description of TERM-FAIL-TYPE errors:

Error	Value	Description
TERM-FAIL-DB2	P	A DB2 error occurred while accessing the RPROF (profile) table.
TERM-FAIL-IEC	I	An internal Gen error occurred in the Dialog Manager.
TERM-FAIL-EXEC	E	A database error occurred in an action block or procedure.
TERM-FAIL-DIALOG	D	A non-database error occurred in the Dialog Manager.
TERM-FAIL-TSQ	Q	An error occurred while accessing the CICS temporary storage queue profile table.

Remaining Fields

The remaining CBLTERM fields are described in the table:

Field	Description
TERM-ERROR-ACTION-NAME	Contains the name of the action block.
TERM-DEFAULT-MSG	This is an output field that by default contains the following message: TIRM000E: SYSTEM ERROR OCCURRED - CONTACT SUPPORT The message can be changed in the termination exit to anything meaningful to the user. For online procedures with a screen, the message is visible in the error message field when the screen is redisplayed.
TERM-SYSTEM-PRINTER	Printer TERMID if the action block executed a PRINTER TERMINAL IS statement.
TERM-ERROR-ENCOUNTERED-SW	Indicates the message: TIRM037E: ** A FATAL ERROR HAS BEEN ENCOUNTERED **

Field	Description
TERM-VIEW-OVERFLOW-SW	Indicates the message: TIRM037E: ** FATAL VIEW OVERFLOW HAS BEEN ENCOUNTERED **
TERM-ACTION-ID	Is appended to the message: TIRM034E: LAST OR CURRENT DATABASE STATEMENT = ...
TERM-ATTRIBUTE-ID	Is appended to the message: TIRM040E: PERMITTED VALUES MISMATCH, FIELD = F ...
TERM-STATUS-FLAG	Produces the message: TIRM038E: ** FATAL DATABASE ERROR HAS BEEN ENCOUNTERED **
TERM-LAST-STATUS	Is appended to the message: TIRM039E: DB LAST STATUS = ...
TERM-TRACE-PTR	This field is documented in online help under the error message TIRM039E.
TERM-LAST-STATEMENTNUM	Is appended to the message: TIRM035E: CURRENT STATEMENT BEING PROCESSED = ...
TERM-CURR-AB-ID	Is appended to the message: TIRM032E: LAST OR CURRENT ACTION BLOCK ID = ...
TERM-CURR-AB-NAME	Is appended to the message: TIRM033E: LAST OR CURRENT ACTION BLOCK NAME = ...
TERM-EABPCB-CNT, TERM-EABPCB-ENTRY, TERM-EABPCB-PTR	These fields describe PCB pointers. The first is the IO-PCB, the second is the ALTERNATE-IOPCB; the last is a database pointer.

Field	Description
TERM-SQLCA-PTR	Pointer to the SQLCA. To address the fields in SQLCA, first define it in the Linkage Section. Use the following example: MY-SQLCA FILLER MY-SQL-CODE FILLER Then add a SET statement at the beginning of the procedure division as shown : SET ADDRESS OF MY-SQLCA TO TERM-SQLCA-PTR
TERM-IEF-COMMAND	The special field of COMMAND.
TERM-IEF-TRANCODE	The special field of TRANCODE.
TERM-EXIT-STATE	The exit state number.
TERM-EXIT-INFOMSG	The exit state message.
TERM-USER-ID	The special field of USERID.
TERM-TERMINAL-ID	The special field of TERMID.
TERM-PRINTER-ID	Represents the ID of the system printer.
TERM-DIALOG-MESSAGENUM	The message number is the FAIL-MSG-NO set be the Dialog Manager. See the Messages Guide for the message represented by the error code displayed.
TERM-OUTPUT-MESSAGE	Before TIRFAIL calls TIRTERMA, it prepares a table of messages that will display on return from the exit if the TERM-STATUS-CODE is a space or a zero. These messages are available to the exit. The last line with a message is followed by a line of all spaces.
TERM-DIALECT-NAME	The current dialect

Field	Description
TERM-FAILURE-MESSAGE-TEXT	The text of the failure message. This may be moved to TERM-DEFAULT-MSG if you want it displayed on the application screen instead of the message: TIRM000E: SYSTEM ERROR OCCURRED - CONTACT SUPPORT

Return Code

TIRTERMA can return three valid status codes to the fail routine. They are used to control what screens will be displayed to the user. For example, if you want to replace the CA Gen error screen with your own and then return to the application screen, and you can modify the code to return the application screen, you can modify the code to return status code 1.

The following table provides a description:

Status Code	Error Message Screen Displayed	Application Screen Displayed
1	No	Yes
2	No	No
0, blank or other (except 1 or 2)	Yes	Yes

The skeleton exit contains example code for each of these status codes with the code for ‘1’ and ‘2’ commented out.

Default Processing

If a runtime error occurs and the default termination exit is used, processing is as follows:

1. The Dialog Manager performs all necessary rollbacks. This is done regardless of the termination exit used.
2. The Dialog Manager fail routine calls the default termination exit. It returns to the fail routine without doing anything, which causes the default termination logic in the fail routine to be used.

3. The CA Gen fail routine displays an error screen that lists the appropriate CA Gen runtime error messages. See the following error message screen:

```
TIRM030E: APPLICATION FAILED ** UPDATES HAVE BEEN BACKED OUT

TIRM031E: FAILING PROCEDURE EXIT DATA FOLLOWS

TIRM032E: LAST OR CURRENT ACTION BLOCK ID = 507774696

TIRM033E: LAST OR CURRENT ACTION BLOCK NAME = ABADDEMP

TIRM034E: LAST OR CURRENT DATABASE STATEMENT =

TIRM035E: CURRENT STATEMENT BEING PROCESSED = 10

TIRM037E: ** A FATAL ERROR HAS BEEN ENCOUNTERED **

TIRM046E: *** TRANSACTION PROCESSING TERMINATED

TIRM044E: *** PRESS PA2 TO CONTINUE ***
```

4. When you press PA2 (NEXT PAGE key) from the error message screen, CA Gen displays the last screen for the transaction that was being processed when the error occurred.

   If you are using the Testing Facility, the PA2 key is the ISPF NEXT PAGE key you defined on the Test Environment Panel.

5. CA Gen recovers all data in the import views at the time the error occurred. Therefore, all user input is recovered and displayed upon the screen. Screen fields that are only in the export view may or may not be populated, depending on when the error occurred.

6. An error message appears in the system error message area defined for the screen. This message is distinct from the runtime error messages displayed on the error message screen. The default error message is:

   SYSTEM ERROR OCCURRED - CONTACT SUPPORT.

   See the following illustration for an example of an application screen that is displayed after an error has occurred.

7. The transaction is terminated, but the application remains active and the user can continue with another transaction as shown in the following screen:

```
IEFSLSB                              CORPORATE MANAGEMENT

                                     EMPLOYEE MAINTENANCE

EMPLOYEE NUMBER: 123456                                    NAME: MICHAEL
WILSON

COST CENTER: 123                                           DEPARTMENT: 4

EMPLOYMENT DATE: 082596                                    STATUS: E

SALARY: 1234

ADDRESS: 7250 MICHIGAN                                     PHONE: (214)
555-1414

CITY/STATE/ZIP: PARIS, TEXAS 73000                         BIRTH DATE:
051067

F02=HELP F05=MAINMENU F07=ADDEMP2

TIRM000E: SYSTEM ERROR OCCURRED - CONTACT SUPPORT
```

Customizing the Exit

Copy Member TIRTERMA from the CA Gen CEHBSAMP library to one of your libraries.

Unlike the other user exits, the skeleton exit is not the source for the default exit in the load library. In prior releases of CA Gen, the termination exit was written in Assembler Language and named TIRTERM. The default exit TIRTERMA in the load library calls TIRTERM for compatibility with prior releases.

When you have completed your modifications, install the exit.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRHELPX—Help Interface Exit

TIRHELPX is a runtime exit containing code for the flow to the application Help business system. TIRHELPX is written in COBOL. The same sample exit is included in various z/OS Runtime DLLs, each used by the specific target environment.

Sample code is included as comments. Until modified, TIRHELPX displays the message:

```
No help available.
```

If the TIRHELPX sample code is used (by removing the comment characters), the following occurs:

1. TIRHELPX checks the flags and determines which procedure in the application help business system (field description, screen description, or permitted values) to flow to.
2. TIRHELPX then calls TIRMSG, which performs the transfer to the target procedure.
3. After the help procedures have completed and the exit is entered, a SET NEXTTRAN TO CONCAT(TRIM(input tirhelp trancode) is performed to return to the calling application.

Source Code

TBD

Purpose

The Dialog Manager invokes the Help exit whenever a command equals HELP or PROMPT (or a synonym translates to HELP or PROMPT).

TIRHELPX handles these commands as shown in the following table:

Condition	Screen Message
HELP command issued while the cursor is not on a field	No help available for this screen
HELP command issued while the cursor is on a field	No help available for this field
PROMPT command issued while the cursor is not on an enterable field	Prompt is valid only for enterable fields
PROMPT command issued while the cursor is on an enterable field	No Prompt data available for this field

CA Gen provided code, to invoke the application help system, is included in this module as comments. The comments may be uncommented and used as an example of how to invoke a user-supplied help system.

By customizing TIRHELPX, you can access a help system that is:

- A user-written system
- An application developed using CA Gen, with the features of unformatted input and PAD NEXT TRAN providing flows
- A third-party help system

Direct use of the Central Encyclopedia as tables for a help system is not supported. However, data needed for the tables of a help system could be extracted using the Public Interface.

CA Gen supports invocation of a help system for both screen and field levels. In addition, you can restart the original application with field data from the help system.

Arguments

TBD

Return Code

The following table lists the return codes and the action taken by the Dialog Manager:

Return Code	Action Taken by Dialog Manager
NM	Transaction terminated and screen not redisplayed (TIRHELPX handles help display).
WM	Screen redisplayed with error message (transaction terminates normally).
F	Termination exit invoked to display severe error messages (fatal errors).

When you have completed your modifications, install the exit.

Default Processing

As provided by CA Gen, the Help exit merely redisplays the screen with a message indicating no help or prompt data is available.

Customizing the Exit

Copy the default exit from the CA Gen CEHBSAMP library to one of your libraries. The member name is TIRHELPX. The skeleton includes sample code, as comments, for invoking the CA Gen Help system. This code may be uncommented for use with that system, or used as an example for invoking other help applications.

The Dialog Manager provides an error message parameter to the Help exit and a parameter list that contains the following information:

- Screen help identifier (helpid). This identifier is specified at the workstation level during screen design. The helpid can be a key into a DB2 table, for instance. For an explanation of the screen help identifier see the Design Guide.
- Current transaction code, screen name, user ID, terminal ID, printer ID, and current dialect.
- Last displayed message and command before requesting help.

If the cursor is on a field that has a field help identifier (helpid), the Dialog Manager also provides the following information:

- Field help identifier (helpid). This identifier is specified at the workstation level during screen design. For an explanation of the screen help identifier see the Design Guide.
- Current, edited values of the field as it appears on the screen, screen length of the field, and a token identifier for the field. These may be passed to the help system and used to replace the contents of the field upon return from the help system.

TIRHELPX calls User Exit TIRMTQB. TIRMTQB finds a message number in the runtime error message table and returns the corresponding message and its length. If the message number is not in the table, TIRMTQB returns a default message. For more information, see Runtime Message Table Exit in this chapter.

TIRHELPX can use the provided data to invoke a user-written help system. TIRHELPX must set a return code to instruct the Dialog Manager which action to take. The Dialog Manager can either terminate the transaction and not redisplay the screen, or redisplay the screen with the contents of the message parameter as the screen's error message.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRIEX—Enhanced Map Input Edit Exit

TIRIEX is provided so that the user can modify the standard CA Gen input editing function for the enhanced map generation mode.

Source Code

This exit is used by Enhanced Map Screens only. The source code for this exit is in CA Gen CEHBSAMP library, in member TIRIEX. The sample exit provided is written in Assembler and uses standard OS Linkage.

The Linkage Parameter list used by TIRIEX is as follows:

```
ARG_RT1            DS    A            I/O PCB OR EIB
ARG_RT2            DS    A            ALT I/O PCB OR COMMAREA
ARG_IEX_COMMAREA   DS    A            PTR TO IEX COMM AREA
ARG_PATTR_DESC     DS    A            PTR TO ATTRIBUTE DESCRIPTOR
ARG_PFIELD_DESC    DS    A            PTR TO FIELD DESCRIPTOR
ARG_PIMAGE_DESC    DS    A            PTR TO IMAGE DESCRIPTOR
ARG_PEP_DESC       DS    A            PTR TO EDIT PATTERN DESC
ARG_PWORK          DS    A            PTR TO 4K WORK AREA
ARGGDTA            DS    A            GLOBDATA
```

Purpose

TIRIEX is provided so that the user can modify the standard Gen input editing function for the enhanced map generation mode. The following types of data are inputs for this exit:

- Date
- Time
- Time Stamp
- Numeric Data
- Text
- Picture (Numeric Text)

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
ARG_RT1	input/ output	Address of the IEB
ARG_RT2	input/ output	Address of COMMAREA

Name	I/O	Description
ARG_IEX_COMMAREA	input/ output	Address of the exit control block
ARG_PATTR_DESC	input/ output	Address of attribute descriptor
ARG_PFIELD_DESC	input/ output	Address of the field descriptor
ARG_PIMAGE_DESC	input/ output	Address of the image descriptor
ARG_PEP_DESC	input/ output	Address of the edit pattern descriptor
ARG_PWORK	input/ output	Reentrant work area
ARGGDTA	input/ output	Address of Globdata

Return Code

No explicit return code is set by the user exit.

Default Processing

The Default Input Edit Exit does not perform any processing.

Customizing the Exit

Modify the exit to perform the specific desired functions using the instructions in the exit source code file.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

TIRIEXS – Standard Map Input Edit Exit

The TIRIEXS exit is called by any blockmode application containing Standard Map screens.

Source Code

The sample source for this exit can be found in member TIRIEXS in the CA Gen CEHBSAMP library. The sample exit is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                PIC X.

01  RUNTIME-PARM2                PIC X.

01  TIRIEXS-CMCB.

   03  TIRIEXS-RETURN-CODE          PIC X.

      88  RC-ACCEPT                VALUE '0'.

      88  RC-REJECT                VALUE '1'.

      88  RC-REPROCESS             VALUE '2'.

      88  RC-EXIT-VALUE            VALUE '3'.

      88  RC-ERASE                 VALUE '4'.

   03  TIRIEXS-STATUS-CODE1         PIC X.

      88  SC-OK                    VALUE ' '.

      88  SC-FAIL-PENDING          VALUE 'F'.

   03  TIRIEXS-STATUS-CODE2         PIC X.

      88  SC-FIRST-PASS            VALUE '1'.

      88  SC-REENTER               VALUE '2'.

   03  TIRIEXS-ERROR-MSG-NUMBER     PIC S9(4) COMP

   03  INPUT-VALUE                  PIC X(256).

   03  INPUT-VALUE-CHAR

         REDEFINES INPUT-VALUE

         OCCURS 256 TIMES          PIC X.

   03  FIELD-LENGTH                 PIC 9(4).
```

```
03  FIELD-FILL-CHAR              PIC X(2).
03  FIELD-BEGIN-ROWCOL           PIC 9(8).
03  FIELD-END-ROWCOL             PIC 9(8).
03  ATTRIBUTE-VALUE              PIC X(256).
03  ATTRIBUTE-VALUE-CHAR
      REDEFINES ATTRIBUTE-VALUE
      OCCURS 256 TIMES          PIC X.
03  ATTRIBUTE-LENGTH             PIC 9(4).
03  ATTRIBUTE-TYPE               PIC X.
  88  ATTR-TEXT                  VALUE 'T'.
  88  ATTR-VARCHAR               VALUE 'V'.
  88  ATTR-NUMERIC               VALUE 'N'.
03  ATTRIBUTE-DECIMAL-PLACES     PIC 9(2).
03  ATTRIBUTE-CASE-SENSITIVE     PIC X.
03  MAPNAME                      PIC X(8).
03  MODNAME                      PIC X(8).
03  DIALECT-NAME                 PIC X(8).
03  DECIMAL-INDICATOR            PIC X.
03  THOUSANDS-INDICATOR          PIC X.
03  CURRENCY-INDICATOR           PIC X.
03  TXT-ORIENTATION              PIC X.
03  NUM-ORIENTATION              PIC X.
03  EDIT-PATTERN-CLASS           PIC X.
  88  EPAT-ALPHANUMERIC          VALUE 'T'.
  88  EPAT-NUMERIC               VALUE 'N'.
```

```
        88  EPAT-DATE              VALUE 'D'.
        88  EPAT-TIME              VALUE 'M'.
        88  EPAT-TIMESTAMP         VALUE 'Q'.
        88  EPAT-NONE              VALUE ' '.
    03  FILLER                 PIC X(100).
01          GLOBDATA                size 3645 bytes.
```

Purpose

TIRIEXS is provided to allow customization of the input editing behavior for Standard Map screens. This exit is used by any blockmode application containing Standard Map screens and is called for each input screen field.

Arguments

The following table gives a brief description of each of the arguments.

Name	Input/ Output	Description
RUNTIME-PARM1	input	DFHEIBLK (CICS) I/O PCB (IMS) Emulated I/O PCB (TSO)
RUNTIME-PARM2	input	DFHCOMMAREA (CICS) Alternate I/O PCB (IMS) Emulated Alt I/O PCB (TSO)
TIRIEXS-CMCB	input/output	A structure containing the following items:

Name	Input/ Output	Description
TIRIEXS-RETURN-CODE	output	The return value indicating what action the exit wants the runtimes to take for the current screen field. RC-ACCEPT - The exit took no action for the current screen field and accepts the results of Gen's validation. Normal processing will continue. RC-REJECT - The exit requests that the value input in the current screen field be marked in error. This return code should only be used for screen fields that have an edit pattern defined or for screen fields mapped to view attributes that are mandatory or that have permitted values defined. RC-REPROCESS - The exit modified the value that was input in the current screen field (INPUT-VALUE) and requests that this modified input value be revalidated. RC-EXIT-VALUE - The exit modified the value that was stored in the view attribute mapped to the current screen field (ATTRIBUTE-VALUE) and requests that this modified attribute value replace the value that was determined by Gen. This modified attribute value will not be revalidated. RC-ERASE - The exit requests that the value input in the current screen field be erased. This return code should not be used for screen fields mapped to mandatory view attributes.

Name	Input/ Output	Description
TIRIEXS-STATUS-CODE1	input	The status of the validation performed by Gen for the current screen field. SC-OK - The current screen field's value is considered to be valid by Gen's validation routines. SC-FAIL-PENDING - The current screen field's value is considered to be invalid by Gen's validation routines.
TIRIEXS-STATUS-CODE2	input	The processing status of the current screen field. SC-FIRST-PASS - This indicates this is the first time the exit has been called for the current screen field. SC-REENTER - This indicates the exit has been called previously for the current screen field and the exit requested that the field be reprocessed (RC-REPROCESS).
TIRIEXS-ERROR-MSG-NUMBER	input	The error message number determined by Gen's validation routines prior to calling the exit.
INPUT-VALUE	input/output	The value entered in the current screen field. This value should be modified by the exit if the exit returns RC-REPROCESS.
FIELD-LENGTH	input	The length of the current screen field.
FIELD-FILL-CHAR	input	The fill character defined for current screen field.
FIELD-BEGIN-ROWCOL	input	The beginning row and column of the current screen field.
FIELD-END-ROWCOL	input	The ending row and column of the current screen field.
ATTRIBUTE-VALUE	input/output	The value to be stored in the view attribute mapped to the current screen field. This value should be modified by the exit if the exit returns RC-EXIT-VALUE.

Name	Input/ Output	Description
ATTRIBUTE-LENGTH	input	The length of the view attribute mapped to the current screen field.
ATTRIBUTE-TYPE	input	The datatype of the view attribute mapped to the current screen field. ATTR-TEXT - A fixed-length text attribute. ATTR-VARCHAR - A varying-length text attribute. ATTR-NUMERIC - A numeric attribute.
ATTRIBUTE-DECIMAL-PLACES	input	The number of decimal places defined for the view attribute mapped to the current screen field.
ATTRIBUTE-CASE-SENSITIVE	input	The case sensitivity property defined for the view attribute mapped to the current screen field.
MAPNAME	input	The name of the current screen.
MODNAME	input	The modname of the current screen.
DIALECT-NAME	input	The name of the dialect used by the current screen.
DECIMAL-INDICATOR	input	The character used to represent the decimal place in the current dialect.
THOUSANDS-INDICATOR	input	The character used to represent the thousands separator in the current dialect.
CURRENCY-INDICATOR	input	The character used to represent the currency symbol in the current dialect.
TXT-ORIENTATION	input	The orientation of text fields in the current dialect.
NUM-ORIENTATION	input	The orientation of numeric fields in the current dialect.

Name	Input/ Output	Description
EDIT-PATTERN-CLASS	input	The edit pattern class for the current screen field. EPAT-ALPHANUMERIC - An alphanumeric edit pattern. EPAT-NUMERIC - A numeric edit pattern. EPAT-DATE - A date edit pattern. EPAT-TIME - A time edit pattern. EPAT-TIMESTAMP - A timestamp edit pattern. EPAT-NONE - No edit pattern is defined for the current screen field.
FILLER	input	Filler, for future use
GLOBDATA	input	Global data, used internally

Return Code

Update TIRIEXS-RETURN-CODE with the relevant value.

Default Processing

The default processing of this exit is to take no action for the current screen field and return RC-ACCEPT in TIRIEXS-RETURN-CODE.

Customizing the Exit

Copy the TIRIEXS exit to one of your libraries and modify the exit to perform the desired input editing behavior.

Building on z/OS

For information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

z/OS Blockmode User Exits—IMS

TIRTIARX—DB2 Message Exit

z/OS Dialog Managers use the CA Gen DB2 Message Exit.

Source Code

The source code for the version of the exit used by IMS application is in CA Gen CEHBSAMP library, in member TIRITIAX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRTIARX is as follows:

```
01  RUNTIME-PARM1                PIC X.
01  RUNTIME-PARM2                PIC X.
01  TIRFAIL-SQLCA                PIC X.
01  TIRTIAR-ERRORS               PIC X.
01  TIRTIAR-TEXT-LEN             PIC X.
01  TIRTIAR-WORKAREA            PIC X.
01  GLOBDATA                    structure size 3645 bytes.
```

Purpose

The DB2 Message Exit is used by all applications targeting DB2 database on z/OS. The TIRFAIL subroutine of the Dialog Manager calls the DB2 Message exit, TIRTIARX, whenever an unrecoverable DB2 failure occurs. TIRTIARX then calls the subroutine DSNTIAR to convert the SQL code into text. The messages returned by DSNTIAR are then merged with the runtime error messages.

TIRTIARX can be customized to statically link DSNTIAR with the executable load module rather than dynamically linking it.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is IO-PCB automatically included if translated.
RUNTIME-PARM2	input	This is ALT-IO-PCB automatically included if translated.
TIRFAIL-SQLCA	input	SQLCA
TIRTIAR-ERRORS	input/output	Error message lines.
TIRTIAR-TEXT-LEN	input	Length of one error message line.
TIRTIAR-WORKAREA	input	Workarea.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Behavior

As provided by CA Gen, the default exit dynamically calls DSNTIAR and is compatible with prior releases. However the sample code also contains examples of how to call DSNTIAR or DSNTIAC statically.

The call to TIRTIARX is made when TIRFAIL is building the table of messages and occurs prior to calling the default termination exit. For more information, see the Online Termination Exit and Batch Termination Exit.

Customizing the DB2 Message Exit

Copy the default exit to one of your own libraries. The member name for IMS is TIRITIAX. The default exit includes example code for the two possible combinations of calls. There are dynamic and static calls of DSNTIAR. Simply comment out the default call and remove the comments from the one you want to use.

When you have completed your modifications, install your exit.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRIRTRX—Default Retry Limit Exit

z/OS Dialog Managers for IMS and IEFAE blockmode applications use the CA Gen Default Retry Limit Exit.

Source Code

This exit is used by IMS and IEFAE blockmode applications only. The source code for this exit is in CA Gen CEHBSAMP library, in member TIRIRTRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRIRTRX is as follows:

```
01  RUNTIME-PARM1                          PIC X
01  RUNTIME-PARM2                          PIC X.
01  RETRY-TIMES                            PIC S9(4) COMP.
01  GLOBDATA                               size 3645 bytes.
```

Purpose

This exit is called at the beginning of a CA Gen IMS or IEFAE blockmode application to enable the defined default value for the TRANSACTION RETRY LIMIT system attribute to be modified.

TRANSACTION RETRY LIMIT will be initialized to this value at the beginning of each new transaction. This value may subsequently be modified by a SET TRANSACTION RETRY LIMIT statement in an action diagram. TRANSACTION RETRY LIMIT is used to specify the maximum number of times a transaction is to be retried when one of the following events occurs:

- A RETRY TRANSACTION action diagram statement executes.
- A deadlock or timeout occurs trying to access a database, and no WHEN DATABASE DEADLOCK OR TIMEOUT statement was provided for that entity action statement. (This is not applicable for z/OS transactions running under IMS. An application running under IMS that encounters a deadlock or timeout will be terminated immediately by IMS itself, even if it has a WHEN DATABASE DEADLOCK OR TIMEOUT statement provided.)

In either of these cases, any uncommitted database updates will be rolled back, and an attempt will then be made to execute the application again. Once the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, no more retries can occur, and the application will fail with a runtime error if the last retry attempt was unsuccessful.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is IO-PCB automatically included if translated.
RUNTIME-PARM2	input	This is ALT-IO-PCB automatically included if translated.
RETRY-TIMES	input/ output	The maximum number of times the transaction execution is retried.

Name	I/O	Description
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

The default processing of this exit is not to modify the Default Retry Limit Exit, that is set to 10. If the Default Retry Limit Exit is used, it must not return a value greater than that specified in the Ultimate Retry Limit Exit.

Customizing the Exit

Modify the source code to set the RETRY-TIMES to the number of retries the applications should use.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRIURTX—Ultimate Retry Limit Exit

z/OS Dialog Managers for IMS blockmode applications use the CA Gen Ultimate Retry Limit Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library in member TIRIURTX. The sample exit is written in COBOL and uses standard OS linkage.

The Linkage Parameter list used by TIRIURTX is as follows:

```
01  RUNTIME-PARM1               PIC X.
01  RUNTIME-PARM2               PIC X.
01  ULTIMATE-RETRY-LIMIT        PIC S9(9)   COMP.
01  GLOBDATA                     size 3645 bytes.
```

Purpose

The Ultimate Retry Limit Exit is used by all applications targeting DB2 database on z/OS. The Ultimate Retry Limit Exit allows the user to specify a maximum value for the TRANSACTION RETRY LIMIT system attribute. This value may never be exceeded, either by a SET TRANSACTION RETRY LIMIT statement in an action diagram, or by the Default Retry Limit Exit.

For an explanation of when and how the TRANSACTION RETRY LIMIT system attribute is used see Default Retry Limit Exit in this chapter.

This exit provides a safeguard in case the system attribute TRANSACTION RETRY LIMIT is set to an excessive value by an action diagram. Once the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, no more retries can occur, and the application will fail with a runtime error if the last retry attempt was unsuccessful.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	Input	This is IO-PCB automatically included if translated.
RUNTIME-PARM2	Input	This is ALT-IO-PCB automatically included if translated.
ULTIMATE-RETRY-LIMIT	Input/ output	The absolute limit which is defaulted to 99.
GLOBDATA	Input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

If the Ultimate Retry Limit Exit is not modified, the maximum value of TRANSACTION RETRY LIMIT will be 99. The Ultimate Retry Limit Exit may be modified to return a value of zero to suppress all retry attempts.

Customizing the Exit

Copy TIRIURTX exit to one of your libraries or directories. For TSO and IEFAE; applies only when RETRY TRANSACTION statement executes.

Modify the copied exit as needed. When you have completed your modifications, install the exit as described in Customizing and Installing z/OS User Exits.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRSYSIX—System ID Exit

z/OS Dialog Managers use the CA Gen System Identification Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library in member TIRTSYSX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRSYSIX is as follows:

```
LINKAGE SECTION.
01  RUNTIME-PARM1        PIC X.
01  RUNTIME-PARM2        PIC X.
01  LOCAL-SYSTEM-ID      PIC X(8)
```

Purpose

This exit is called by all IMS applications.The purpose of TIRSYSIX is to enable logic that lets the same application be implemented on multiple systems and perform processing specific to each system targeted.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
IO-PCB	input	Automatically included if translated.

Name	I/O	Description
ALT-IO-PCB	input	Automatically included if translated.
LOCAL-SYSTEM-ID	output	The identifier of the system where the application is executing.

Return Code

No explicit return code is set by the user exit.

Default Processing

The literal IMS is returned.

Customizing the Exit

Copy the TIRCSYSX member to one of your libraries and modify to populate the LOCAL-SYSTEM-ID as required by the application.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRALLOX
- TIRPROUX

TIRUSRIX—User ID Exit

z/OS Dialog Managers use the CA Gen User Identification Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library in member TIRIUSRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRIUSRX is as follows:

```
01  IO-PCB.
    03  IO-PCB-LTERM                   PIC X(8).
    03  FILLER                         PIC X(2).
    03  IO-PCB-STATUS                  PIC X(2).
    03  IO-PCB-INPUT-PREF.
        05  IO-PCB-DATE                PIC S9(7) COMP SYNC.
        05  IO-PCB-TIME                PIC S9(7) COMP SYNC.
        05  IO-PCB-MSG-SEQ             PIC S9(7) COMP.
    03  IO-PCB-MAPNAME                 PIC X(8).
    03  IO-PCB-USER-ID.
        05  IO-PCB-USER-ID-C1          PIC X.
        05  FILLER                     PIC X(7).
01  ALT-IO-PCB                         PIC X.
01  FILLER-PARM                        PIC X.
01  TIRUSRID-PARM.
    05  IET-USER-ID                    PIC X(8).
    05  IET-USER-ID2                   PIC X(8).
```

Purpose

This exit is called by all IMS applications. The purpose of TIRUSRIX is to obtain the userid and terminal ID of the executing application so that these values can be used as part of the key for the DB2 Profile Table and in the application itself.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description		
I/O-PCB	input	This is IO-PCB automatically included if translated.		
ALT-IO-PCB	input	This is ALT-IO-PCB automatically included if translated.		
FILLER-PARM	input	Not used.		
TIRUSRID-PARM	output	A pointer to a structure containing the following items:		
		IET-USER-ID	output	The userid to be used in the application.
		IET-USER-ID2	output	The ID to be used as part of the Profile Table key.

Return Code

No explicit return code value is set by the user exit.

Default Processing

If the user ID from the IO-PCB is valid the exit returns its value for both fields otherwise the terminal ID from the IO-PCB is returned for both fields.

Customizing the Exit

Copy the TIRUSRIX to one of your libraries and modify to populate either IET-USER-ID or IET-USER-ID2 as required by the application.

Note: IET-USER-ID is used by the application as its User Identifier while IET-USER-ID2 is used as part of the Key to the RPROF (Profile Manager) Table.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

TIRSECRX

TIRSECRX—Security Interface Exit

z/OS Dialog Managers use the CA Gen Security Interface Exit.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRSECRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                          PIC X.
01  RUNTIME-PARM2                          PIC X.
01  TIRSECR-CMCB.
    03  TIRSECR-USERID                     PIC X(8).
    03  TIRSECR-TRANCODE                   PIC X(8).
    03  TIRSECR-TERMINAL-ID                PIC X(8).
    03  TIRSECR-SYSTEM-ID                  PIC X(8).
    03  TIRSECR-LOAD-MODULE                PIC X(8).
    03  TIRSECR-PSTEP-NAME                 PIC X(32).
    03  TIRSECR-DIALECT                    PIC X(32).
    03  TIRSECR-RETURN-CODE                PIC XX.
    03  TIRSECR-FAILURE-MSG                PIC X(80).
01 GLOBDATA                                size 3645 bytes.
```

Purpose

This exit is used by all IMS applications. The purpose of the TIRSECRX exit is to allow transaction-level security checking to be implemented. The Dialog Manager calls the Security Interface Exit when a transaction is started and before execution of a dialog flow. This allows transaction-level security checking to be implemented. After it has been enabled, the Dialog Manager executes the security interface exit automatically, at the relevant points, without any intervention by a programmer, when invoking any load modules in a business system.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is IO-PCB automatically included if translated.
RUNTIME-PARM2	input	This is ALT-IO-PCB automatically included if translated.
TIRSECR-CMCB	input/ output	A structure containing the following items:
	TIRSECR-USERID	input — The userid under which this transaction is executing, as provided by the TIRUSRIX exit.
	TIRSECR-TRANCODE	input — The load module transaction code.
	TIRSECR-TERMINAL-ID	input — The terminal ID used by this transaction, spaces if this is a non-terminal transaction.

Name	I/O	Description
	TIRSECR-SYSTEM-ID	input	The system ID where this transaction is executing, as provided by the TIRSYSIX exit.
	TIRSECR-LOAD-MODULE	input	The load module name.
	TIRSECR-PSTEP-NAME	input	The Procedure Step name.
	TIRSECR-DIALECT	input	The dialect used by this application.
	TIRSECR-RETURN-CODE	output	A 2-byte character field returning the result of the security check. The following values are supported: SPACES—TIRSECR-ALL-OK Anything else—failure
	TIRSECR-FAILURE-MSG	output	An 80-byte character field, to be populated by this exit, to describe the failure with a message of choice.

Return Code

Update TIRSECR-RETURN-CODE with the relevant value.

Default Processing

The default processing of this exit is to do no security checking and to return TIRSECR-ALL-OK as the return code.

Customizing the Exit

Copy the TIRSECRX exit to one of your libraries and modify to perform security checking as required by the application. Ensure that TIRSECR-RETURN-CODE is set to spaces when the security check is successful or some other value to indicate failure. If a message describing the violation is returned in TIRSECR-FAILURE-MSG, the Dialog Manager will pass it to TIRTERMA.

Building on z/OS

For information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRUSRIX
- TIRSECVX
- TIRELOGX
- TIRTERMA

TIRDATX—Date and Time Services Exit

z/OS Dialog Managers use the CA Gen Date and Time Services Exit.The Date and Time Services Exit can be used to intercept, adjust, or validate system dates and times. This exit is provided to allow user modification and customization of date and time processing.

Source Code

The source code for this exit is in the Gen CEHBSAMP library in member TIRDATX. The sample exit provided is written in Assembler and uses standard OS Linkage.

The parameter list used by TIRDATX is as follows:

```
PARMRT1  DS    A
PARMRT2  DS    A
PARMCMCB DS    A
PARMWORK DS    A
PARMGDTA DS    A
```

Purpose

This exit receives control for some but not all date and time services. Only services that acquire, or manipulate the date and time, where that date or time was acquired from the system, or where validation is involved, invoke this exit.

This exit is not invoked for the following conditions:

- Services involving conversion from one form to another does not invoke this exit.
- If some error condition exists. For example, if the clock is not set, the date and time services return the error directly to the requester and do not call this exit.

- For validation, if the value is not valid, the failure is returned to the requester and the exit is not called.
- If this exit changes a date or time and requests re-validation, and the value is in error, the error is returned to the requester and the exit is not called.

Note: If the date and time is modified by the exit, the exit must indicate this by returning the appropriate return code. Return codes that are invalid (not one of the listed values) will be ignored and the result is as if the exit returned zero (0). Therefore it is imperative that you not take advantage of any behavioral aspects not explicitly documented here or in the sample code since future releases could change the operation.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
PARMRT1	input	This is IO-PCB automatically included if translated.
PARMRT2	input	This is ALT-IO-PCB automatically included if translated.
PARMCMCB	input/ output	Address of the Date Communication Control Block
PARMWORK	input/ output	Address of a 256 byte workarea
PARMGDTA	input	Address of GLOBDATA

Return Code

Only some of the date and time services functions are available to this exit, these are indicated by the value of the DREQ field. Before returning this exit must restore registers 0-14 to their values on entry and update register 15 with the return code that controls how the date and time services functions continue processing.The return code varies depending on the request, as follows:

DREQ - Service Request DREQVAL

- 0—Use the system date/time as stored in DCMCB. The exit has not modified these values and accepts them as they are.
- 4—The exit has modified the date/time stored in the DCMCB and requests that the validation be re-executed for these values.

  Note: The exit will be called again after validation is complete.

- 8—The exit has modified the date/time stored in the DCMCB and does not require the values be re-validated. The modified values are returned to the requester.
- 12—The exit requests that the date/time service fail the request. This is returned to the service requester as if the date and/or time were invalid.

DREQ - Service Request DREQSD, DREQAS, DREQVTS, and DREQST

- 0—Use the system date/time as stored in DCMCB.
- 4—The exit requests the date, time, or timestamp value be recomputed. If the exit has modified any of these values, the modifications are discarded and the values computed from the system clock. For DREQAS, the DINC value represents the number of days to be added to the date. The exit is called again after the date and time have been recomputed.
- 8—The exit has modified the date/time stored in the DCMCB and does not require the date/time services recompute the associated values. The modified and unmodified values are returned to the service requester unchanged.
- 12—The exit requests that the date/time service fail the request. This is the same as if the clock was not set or was damaged.

Default Processing

Sample Code

Sample code has been provided as commented out blocks of code. This is as an example only. To use the sample code, you must remove the comments. Sections that are specific to a particular system, such as IMS, are indicated by comments preceding and following the code. Common code that should be used by both examples is also indicated.

The sample code provides the ability to respond to request code 1 (get system date) and request code 7 (get current timestamp). The date information is read from a file and is used to change the year, month, and day. The timestamp information (hours, minutes, seconds, microseconds) is read and left unchanged.

Delivery Configuration

As configured, the sample code will read the required date from a file when generated for IMS. This exit can also obtain values from DB2 table lookups.

To use this facility, you must change the source code to set a local variable as appropriate. If IMS mode, use the assembler but do not use the preprocessor.

Registers

Register 14 contains the address that control is to be returned to, and Register 13 contains the address of a save area set up for the exit's usage. All registers must be saved on entry. Register 15 must be updated with the return code and all other registers must be saved and restored on return.

Customizing the Exit

You can customize the exit to perform your specific needs. The following paragraphs provide guidelines to be observed when modifying this exit. Be sure to read all notes provided with the sample code for the latest information on using this exit.

DREQ Service Request

This exit uses operating system standard linkage. On return, registers 0 - 14 must be restored to their values on entry. Register 15 contains a return code to control the processing of the date and time services. The service request code is indicated in the Date CMCB field, DREQ. The return codes and service requests are discussed later in this section.

I/O Format

The format of input and output data are indicated in the CMCB fields DDATEF and DTIMEF. These values should be examined to determine the format of the data to be stored, or to be used as input by the exit.

Fields in the Date CMCB

Other fields in the CMCB have various meanings and formats as described in the following paragraphs.

DDATE

This field contains the binary date value. It is treated as a signed decimal number and converted to binary. The format is specified by the field DDATEF and cannot be changed. This field can be in one of the following formats:

- YYYYMMDD—Four digit year, two digit month, and two digit day
- YYMMDD—Two digit year (the century is omitted), two digit month, and two digit day
- CYYMMDD—One digit century code, two digit year, two digit month, and two digit day

Note: The one digit century code (C) is a number from 0 to 9, inclusive. The century ranges that can be represented are from 1600 to 2599, inclusive. The century codes are: 0 = 19XX, 2 = 20XX, 3 = 22XX, 4 = 23XX, 5 = 24XX, 6 = 25XX, 7 = 16XX, 8 = 17XX, and 9 = 18XX.

DDATEF

This field contains an indicator of the format of the DDATE field's content and cannot be changed.

DTIME

This field contains the binary time value. The time is treated as a signed decimal number with the format HHMMSSTH, or HHMMSST, or HHMMSS with the following conventions:

- HH—Hours
- MM—Minutes
- SS—Seconds
- T—Tenths of seconds
- H—Hundredths of seconds

The format used is specified by the DTIMEF field.

DTIMEF

This field contains an indicator of the format of the DTIME field's content and cannot be changed.

DTSTAMP

This field contains the zoned decimal time stamp value in a fixed format of YYYYMMDDHHMISSNNNNNN with the following conventions:

- YYYY—Four-digit year
- MM—Two-digit month
- DD—Two-digit day
- MI—Two-digit minutes
- SS—Two-digit seconds
- NNNNNN—Six-digit microseconds

DINC

This field contains a signed binary increment to be added to the date value in DDATE when DREQAS service is requested. It is unused in all other cases. A negative value will result in a date prior to the base date.

Testing the DREQ Field

The exit must test the DREQ field of the Date CMCB to determine the service request made of the Date/Time routine. This is used to customize the exit based on your needs. For example, if you wish to perform local validation of dates only, the request of interest is DREQVAL. For all other requests, the exit must return a zero.

Modifying Date and Time

If the exit is used to modify date or time, the exit must modify the appropriate fields for the service request. Different service requests use different areas of the Date CMCB as their input, and place their output in various fields.

Service Requests Intercepted by the Date and Time Services Exit

The service requests intercepted by this exit are:

- DREQVAL—Request date and/or time validation
- DREQSD—Return the current system date and time
- DREQAS—Add a specified increment to the date value
- DREQST—Return the current timestamp
- DREQVTS—Validate the timestamp provided
- DREQST—Return the current system timestamp

DREQ Return Codes

The return codes and their meanings vary for the different service requests indicated in the DREQ field. Refer to the following paragraphs for the request, return codes and meaning.

DREQ - Service Request DREQVAL

- 0—Use the system date/time as stored in DCMCB. The exit has not modified these values and accepts them as they are.
- 4—The exit has modified the date/time stored in the DCMCB and requests that the validation be re-executed for these values. Note that the exit will be called again after validation is complete.
- 8—The exit has modified the date/time stored in the DCMCB and does not require the values be re-validated. The modified values are returned to the requester.
- 12—The exit requests that the date/time service fail the request. This is returned to the service requester as if the date and/or time were invalid.
- DREQ - Service Request DREQSD, DREQAS, DREQVST, and DREQST
- 0—Use the system date/time as stored in DCMCB.
- 4—The exit requests the date, time, or timestamp value be recomputed. If the exit has modified any of these values, the modifications are discarded and the values computed from the system clock. For DREQAS, the DINC value represents the number of days to be added to the date. The exit is called again after the date and time have been recomputed.
- 8—The exit has modified the date/time stored in the DCMCB and does not require the date/time services recompute the associated values. The modified and unmodified values are returned to the service requester unchanged.
- 12—The exit requests that the date/time service fail the request. This is the same as if the clock was not set or was damaged. The following table describes service requests and the fields they use:

Service Request (DREQ)	Input	Output	Applicable Notes
DREQVAL	DDATE, DTIME		1, 4, 5
DREQAS	DDATE, DTIME, DINC	DDATE, DTIME	2, 5
DREQVTS	DREQSD	DDATE, DTIME	2, 5
DREQST		DTSTAMP, DDATE, DTIME	2, 3, 5

Note:

1. Date and/or Time validation can be skipped if the appropriate field is set to zero. For example, if DDATE is zero, then the Date validation is skipped.
2. Initial processing obtains the current date and time using the system clock and adjusts the value based on the time zone adjustment. If the request is DREQAS, then the DINC value is added to the number of days prior to computing the Gregorian date, and then the DDATE / DTIME fields are computed. If the exit requests that the values be reprocessed, any modification that the exit made to the DDATE / DTIME fields is discarded and the values recomputed from the system clock. DINC can be altered if the request was DREQAS.
3. If the request is DREQST, then the system time stamp values are computed from the clock values.
4. Validation returns a code to the requester indicating the validity of the date/time/time stamp. If the value is valid, the exit is called or recalled if the exit requested the validation be reprocessed.
5. The formats of input and output data are indicated in the CMCB fields DDATEF and DTIMEF. These values should be examined to determine the format of the data to be stored, or to be used as input by the exit.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None

TIRDEVI—Device Characteristics Exit

z/OS Dialog Managers use the CA Gen Device Characteristics Exit.

Source Code

The source code for this exit is in the Gen CEHBSAMP library in member TIRDEVI. The sample exit provided is written in Assembler and uses standard OS Linkage.

The parameter list used by TIRDEVI is as follows:

```
PARMRT1  DS    A
PARMRT2  DS    A
EXTATTRA DS    A
DEVUSER  DS    A
DIALECT  DS    A
GLOBDATA DS    A
```

Purpose

This exit is only used by Gen applications that use Standard Mapping facilities, not by Enhanced Map.The Device Characteristics exit, TIRDEVI, is called every time a message is sent from or received by an application. This exit provides the runtime data stream processing routines for the definition of the specific device characteristics.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
IO-PCB	Input	Automatically included by the Translator
ALT-IO-PCB	Input	Automatically included by the Translator
DEVCAP	Input/ output	Device Characteristics
Parm 4	Input/ output	TMOHDLCT (dialect)/workarea
GDTA	Input	Address of GLOBDATA

TMOHDLCT is a pointer to the eight-character dialect name as returned from the User Dialect exit, TIRDLCTX. This value represents the current selected dialect. The default exit returns a default dialect value for this parameter.

The DEVCAP is a pointer to a 256-byte structure defined for the return of the device attributes.

Return Code

The Device Characteristic structure contains the returned device characteristics. The fields in this structure are as follows:

EXTPARM		Returned Device Capabilities
MAXROWR	DS H	(24/32/43/27) maximum number of screen rows
MAXCOLR	DS H	(80/132) maximum number of screen columns
EXTDSR	DS CL1	(0/255) 0= No Extended Data Stream support
EXTCLRR	DS CL1	(0/255) 0= Base Color, 255 = Extended Color
EXTHIGHR	DS CL1	(0/255) 0= No Highlight, 255 = Highlighting
EXTGRID	DS CL1	(0/255) 0= No Grid Line, 255 = Grid Line
EXTDBCS	DS CL1	(0/255) 0= No DBCS DISPLAY or ENTRY
EXTSCS8	DS CL1	(0/255) 0= No DBCS Set F8, SCS'8' for DBCS
XMIXENT	DS CL1	(0/255) 0= No Mixed (SBCS/DBCS) entry
XINEDIT	DS CL1	(0/255) 0= No INPUT EDITING ATTRIBUTE support
XOUTXLAT	DS A	Pointer to 256 byte Output Translate Table
XINPXLAT	DS A	Pointer to 256 byte Input Translate Table
	DS CL235	Filler MUST BE ZERO

Default Processing

The maximum row and column values are derived from the 3270 model type. At this time, CA Gen supports only IBM 3270 model 2 (24 x 80).

Extended Data Stream support and other extended attribute capabilities of the terminal are derived from query or any other user defined method of retrieving the terminal status. If Extended data stream is not enabled, then no extended data stream functions are built into the outbound data stream. If Double Byte Character Support (DBCS) is not enabled, then no DBCS data is placed in the outbound data stream. If MIXENT is not enabled, all mixed entry fields are built as Single Byte Character Support (SBCS) only fields in the data stream.

Additional information is available in the vendor documentation on National Language Support (NLS).

Translate Tables

The output (OUTXLAT) and input (INPXLAT) tables are standard 256 byte translate tables in a format suitable for the translate (TR) operations code (op code). OUTXLAT is used when the current device does not support the same code page as the application and encyclopedia. This means that a difference exists in the code points for the encyclopedia and application database and the code points for the device. The translate table needs to convert the code points in the output data stream to the correct code points for the current device to display the correct glyphs. INPXLAT is used when data is received from the terminal to convert the code points back to the appropriate values for the application database and encyclopedia.

If the device supports the same code page as the application and database, then OUTXLAT and INPXLAT should be set to ZERO (0) to suppress any code point conversion.

For example, if the current device is a Japanese 557x terminal supporting code page 930 (uppercase Roman only) and the application prompts contain lower case Roman letters, the translate tables must perform inbound and outbound translations.

Outbound, the translate table performs monocasing (from lowercase to uppercase), and translates the application database code points to the device code points. This displays the correct glyphs on the device.

Inbound, the translate table translates the device code points to the application database and encyclopedia code points for proper storage. This prevents corruption of the data in the database.

To accomplish the translation process in the preceding example, set OUTXLAT to point to a table that converts lowercase code points to uppercase. Set INPXLAT to a table that translates device Katakana back into the code point values needed in the application database.

Customizing the Exit

Copy the default exit from the CA Gen sample library to a separate library. The member name is TIRDEVI. You can customize this exit to accept input from the User Dialect exit (TIRDLCTX) to change the code page during production.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

TIRDLCTX

TIRDLCTX—User Dialect Exit

z/OS Dialog Managers use the CA Gen User Dialect Exit.

Source Code

The source code for this exit is in the CA Gen CEHBSAMP library member TIRDLCTX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                                PIC X.
01  RUNTIME-PARM2                                PIC X.
01  TIRDLCT-CMCB.
    03  TIRDLCT-USERID                          PIC X(8).
    03  TIRDLCT-TERMINAL-ID                     PIC X(8).
    03  TIRDLCT-SYSTEM-ID                       PIC X(8).
    03  TIRDLCT-RETURN-DIALECT                   PIC X(8).
01  GLOBDATA                                    size 3645 bytes.
```

Purpose

This exit is used by all applications. The purpose of the TIRDLCTX exit is to supply the current user's dialect to the application. It is meaningful for multilingual applications.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is IO-PCB automatically included if translated.
RUNTIME-PARM2	input	This is ALT-IO-PCB automatically included if translated.
TIRDLCT-CMCB	input/ output	A structure containing the following items:

Name	I/O	Description	
	TIRDLCT-USERID	input	The userid under which this transaction is executing, as provided by the TIRUSRIX exit.
	TIRDLCT-TERMINAL-ID	input	The terminal ID used by this transaction, spaces if this is a non-terminal transaction.
	TIRDLCT-SYSTEM-ID	input	The system ID where this transaction is executing, as provided by the TIRSYSIX exit.
	TIRDLCT-RETURN-DIALECT	input	The dialect used by this application.

Return Code

No explicit return code is set by the user exit.

Default Processing

The default processing of this exit is to return a dialect name of DEFAULT.

Customizing the Exit

Copy the default exit from the CA Gen CEHBSAMP library to one of your libraries. The member name is TIRDLCTX. For multilingual support, modify this module to return the appropriate dialect for a user. The dialect returned is the one selected using the Dialect Definition option of the Design Toolset. If none is selected or returned, the default dialect is used.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

TIRDEVI

TIRUPPRX—Uppercase Translation Exit

z/OS Dialog Managers use the CA Gen Uppercase Translation Exit. This exit is also called the Lower-to-Uppercase Conversion Exit.

Source Code

This exit is used by single byte and double byte applications. When used by double byte applications an alternate entry point TIRUPDBx is used. The source code for this for this exit is in CA Gen CEHBSAMP library in member TIRUPPRX. The sample exit is written in COBOL and uses OS linkage.

The Parameter list used by TIRUPPRX is as follows:

```
01  RUNTIME-PARM1                          PIC X.
01  RUNTIME-PARM2                          PIC X.
01  XLATE-TABLE-NAME                      PIC X(8).
01  XLATE-LEN                             PIC S9(4) COMP.
01  XLATE-DATA                            PIC X(4096).
01  GLOBDATA                               size 3645 bytes.
```

Purpose

The purpose of the Uppercase Translation User Exit is to translate character input from lowercase to uppercase. It contains a table of paired lower and uppercase characters. This exit is called by the Dialog Manager to translate the lower caser trancode to upper case, by the TIRFUPPR Function to translate the designated data to upper case and by the Standard Map runtime to translate the identified input data to upper case.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is IO-PCB automatically included if translated.
RUNTIME-PARM2	input	This is ALT-IO-PCB automatically included if translated.
XLATE-TABLE-NAME	input	Name of the translation table to be used.
XLATE-LEN	input	Length of data to be translated.
XLATE-DATA	input/output	Data to be translated.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code value is defined for this exit.

Default Processing

The default processing of this exit is to convert lower case characters to upper case using a table named DEFAULT that contains the English character set(A-Z).

Customizing the Exit

Copy the default exit from the CA Gen CEHBSAMP library to one of your own libraries. The member name is TIRUPPRX.

The exit supports both single byte and double byte languages. Adding support for DBCS is done in the same way as for single byte.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRYYX—Two-Digit Year Input Edit Exit

z/OS Dialog Managers use CA Gen Two-Digit Year Input Exit.

Source Code

The source code for this exit is in the CA Gen CEHBSAMP library member TIRYYX. The sample exit provided is written in Assembler and uses standard OS Linkage.

The parameter list used by TIRYYX is as follows:

```
EXTCB     DS     A
WORKAREA  DS     A
GLOBDATA  DS     A
```

Purpose

This exit is used by CA Gen Standard Map applications only. The purpose of the TIRYYX exit is to process two-digit or YY-style date input and set the century part using any chosen algorithm to implement logic to handle the century part of the date.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
EXITCB	input	Address of the Date Communication Control Block
WORKAREA	input	Address of a 256 byte workarea
GLOBDATA	input	Address of GLOBDATA

Return Code

Before returning this exit must restore registers 0-14 to their values on entry and update register 15 with a value of 4 to indicate that the YY modified by the exit should be used. Any other value, including 0, indicates the original values passed to the exit are acceptable to continue processing.

Default Processing

The exit contains sample code for 2 algorithms but neither are executed. By default the exit returns a value of 0, indicating that no changes were done by the exit.

Customizing the Exit

Copy the TIRYYX exit to one of your libraries.

Internally, CA Gen handles four-digit year dates correctly assuming the user application uses a YYYY edit pattern throughout. If the user interface is designed to accept a two-digit date entry, and defaulting to the current century is not acceptable, use this exit to implement logic to get the required behavior for defaulting the century part of the date. The exit is called to process either a DATE or TIMESTAMP field which utilizes a 2-digit year (YY) in the edit pattern associated with the field. An indicator is set in the exit control block to indicate if the value is a date or timestamp.

Modify the exit to use one of the provided algorithms or add your own as required by your applications.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRTERMA—Termination Exit

The CA Gen Termination Exit is called by z/OS Dialog Managers when a fatal runtime error is encountered.

Source Code

TIRTERMA Termination Exit is used by all non-cooperative applications targeting z/OS. The source code is in CA Gen CEHBSAMP library, in member TIRTERMA. The sample exit provided is written in COBOL and uses standard OS Linkage.

The parameters passed between the fail routine and the termination exits are defined via structure TERM-EXIT-PARM-LIST. This structure is included via copy member CBLTERM, which is also in the CEHBSAMP library.

The Linkage Parameter list used by TIRTERMA is as follows:

```
01  RUNTIME-PARM1              PIC X.
01  RUNTIME-PARM2              PIC X.
01  TERM-EXIT-PARM-LIST        structure defined in CBLTERM.
01  GLOBDATA                   size 3645 bytes.
```

Purpose

The purpose of the TIRTERMA exit is to control how fatal runtime errors are handled by the Dialog Manager.

Runtime errors are either fatal or non-fatal errors. When a non-fatal error occurs, such as invalid user input, the Dialog Manager displays an error message on the transaction screen. You can correct the error and continue processing the transaction.

When a fatal error occurs, transaction processing is terminated. The Dialog Manager executes a fail routine that backs out changes by performing the necessary rollbacks of the databases. The fail routine then calls the termination exit that determines what diagnostic (error) information is displayed and where it is displayed.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.
TERM-EXIT-PARM-LIST	input/ output	Structure of parameters for termination and failure routine. The items in this structure are described in the CBLTERM Field Definitions. Description of these fields follow.
GLOBDATA	input	Global data, used internally.

The structure TERM-EXIT-PARM-LIST is defined via copy member CBLTERM. Two items in this structure control processing. These items are:TERM-STATUS-CODE

When TIRFAIL calls TIRTERMA, TERM-STATUS-CODE is used to control what TIRFAIL does next.

The following table provides a description of each TERM-STATUS-CODE value:

Value	Description
’ ’ (space) or 0 (zero)	TIRFAIL displays the message and redisplays the previous screen with TERM-DEFAULT-MSG in the error message field.

Value	Description
1	This value indicates that TIRTIRMA has handled the messages and will not display them. It will, however, redisplay the previous screen with TERM-DEFAULT-MSG in the error message field.
2	Thisvalue indicates that TIRTIRMA has handled everything. TIRFAIL does not display the messages and does not redisplay the previous screen.

TERM-FAIL-TYPE

The following table contains a description of TERM-FAIL-TYPE errors:

Error	Value	Description
TERM-FAIL-DB2	P	A DB2 error occurred while accessing the RPROF (profile) table.
TERM-FAIL-IEC	I	An internal Gen error occurred in the Dialog Manager.
TERM-FAIL-EXEC	E	A database error occurred in an action block or procedure.
TERM-FAIL-DIALOG	D	A non-database error occurred in the Dialog Manager.
TERM-FAIL-TSQ	Q	An error occurred while accessing the CICS temporary storage queue profile table.

Remaining Fields

The remaining CBLTERM fields are described in the table:

Field	Description
TERM-ERROR-ACTION-NAME	Contains the name of the action block.

Field	Description
TERM-DEFAULT-MSG	This is an output field that by default contains the following message: TIRM000E: SYSTEM ERROR OCCURRED - CONTACT SUPPORT The message can be changed in the termination exit to anything meaningful to the user. For online procedures with a screen, the message is visible in the error message field when the screen is redisplayed.
TERM-SYSTEM-PRINTER	Printer TERMID if the action block executed a PRINTER TERMINAL IS statement.
TERM-ERROR-ENCOUNTERED-SW	Indicates the message: TIRM037E: ** A FATAL ERROR HAS BEEN ENCOUNTERED **
TERM-VIEW-OVERFLOW-SW	Indicates the message: TIRM037E: ** FATAL VIEW OVERFLOW HAS BEEN ENCOUNTERED **
TERM-ACTION-ID	Is appended to the message: TIRM034E: LAST OR CURRENT DATABASE STATEMENT = ...
TERM-ATTRIBUTE-ID	Is appended to the message: TIRM040E: PERMITTED VALUES MISMATCH, FIELD = F ...
TERM-STATUS-FLAG	Produces the message: TIRM038E: ** FATAL DATABASE ERROR HAS BEEN ENCOUNTERED **
TERM-LAST-STATUS	Is appended to the message: TIRM039E: DB LAST STATUS = ...
TERM-TRACE-PTR	This field is documented in online help under the error message TIRM039E.
TERM-LAST-STATEMENTNUM	Is appended to the message: TIRM035E: CURRENT STATEMENT BEING PROCESSED = ...
TERM-CURR-AB-ID	Is appended to the message: TIRM032E: LAST OR CURRENT ACTION BLOCK ID = ...

Field	Description
TERM-CURR-AB-NAME	Is appended to the message: TIRM033E: LAST OR CURRENT ACTION BLOCK NAME = ...
TERM-EABPCB-CNT, TERM-EABPCB-ENTRY, TERM-EABPCB-PTR	These fields describe PCB pointers. The first is the IO-PCB, the second is the ALTERNATE-IOPCB; the last is a database pointer.
TERM-SQLCA-PTR	Pointer to the SQLCA. To address the fields in SQLCA, first define it in the Linkage Section. Use the following example: MY-SQLCA FILLER MY-SQL-CODE FILLER Then add a SET statement at the beginning of the procedure division as shown : SET ADDRESS OF MY-SQLCA TO TERM-SQLCA-PTR
TERM-IEF-COMMAND	The special field of COMMAND.
TERM-IEF-TRANCODE	The special field of TRANCODE.
TERM-EXIT-STATE	The exit state number.
TERM-EXIT-INFOMSG	The exit state message.
TERM-USER-ID	The special field of USERID.
TERM-TERMINAL-ID	The special field of TERMID.
TERM-PRINTER-ID	Represents the ID of the system printer.
TERM-DIALOG-MESSAGENUM	The message number is the FAIL-MSG-NO set be the Dialog Manager. See the Messages Guide for the message represented by the error code displayed.
TERM-OUTPUT-MESSAGE	Before TIRFAIL calls TIRTERMA, it prepares a table of messages that will display on return from the exit if the TERM-STATUS-CODE is a space or a zero. These messages are available to the exit. The last line with a message is followed by a line of all spaces.

Field	Description
TERM-DIALECT-NAME	The current dialect
TERM-FAILURE-MESSAGE-TEXT	The text of the failure message. This may be moved to TERM-DEFAULT-MSG if you want it displayed on the application screen instead of the message: TIRM000E: SYSTEM ERROR OCCURRED - CONTACT SUPPORT

Return Code

TIRTERMA can return three valid status codes to the fail routine. They are used to control what screens will be displayed to the user. For example, if you want to replace the CA Gen error screen with your own and then return to the application screen, and you can modify the code to return the application screen, you can modify the code to return status code 1.

The following table provides a description:

Status Code	Error Message Screen Displayed	Application Screen Displayed
1	No	Yes
2	No	No
0, blank or other (except 1 or 2)	Yes	Yes

The skeleton exit contains example code for each of these status codes with the code for ‘1’ and ‘2’ commented out.

Default Processing

If a runtime error occurs and the default termination exit is used, processing is as follows:

1. The Dialog Manager performs all necessary rollbacks. This is done regardless of the termination exit used.
2. The Dialog Manager fail routine calls the default termination exit. It returns to the fail routine without doing anything, which causes the default termination logic in the fail routine to be used.

3. The CA Gen fail routine displays an error screen that lists the appropriate CA Gen runtime error messages. See the following error message screen:

```
TIRM030E: APPLICATION FAILED ** UPDATES HAVE BEEN BACKED OUT

TIRM031E: FAILING PROCEDURE EXIT DATA FOLLOWS

TIRM032E: LAST OR CURRENT ACTION BLOCK ID = 507774696

TIRM033E: LAST OR CURRENT ACTION BLOCK NAME = ABADDEMP

TIRM034E: LAST OR CURRENT DATABASE STATEMENT =

TIRM035E: CURRENT STATEMENT BEING PROCESSED = 10

TIRM037E: ** A FATAL ERROR HAS BEEN ENCOUNTERED **

TIRM046E: *** TRANSACTION PROCESSING TERMINATED

TIRM044E: *** PRESS PA2 TO CONTINUE ***
```

4. When you press PA2 (NEXT PAGE key) from the error message screen, CA Gen displays the last screen for the transaction that was being processed when the error occurred.

   If you are using the Testing Facility, the PA2 key is the ISPF NEXT PAGE key you defined on the Test Environment Panel.

5. CA Gen recovers all data in the import views at the time the error occurred. Therefore, all user input is recovered and displayed upon the screen. Screen fields that are only in the export view may or may not be populated, depending on when the error occurred.

6. An error message appears in the system error message area defined for the screen. This message is distinct from the runtime error messages displayed on the error message screen. The default error message is:

   SYSTEM ERROR OCCURRED - CONTACT SUPPORT.

   See the following illustration for an example of an application screen that is displayed after an error has occurred.

7. The transaction is terminated, but the application remains active and the user can continue with another transaction as shown in the following screen:

```
IEFSLSB                              CORPORATE MANAGEMENT

                                     EMPLOYEE MAINTENANCE

EMPLOYEE NUMBER: 123456                                   NAME: MICHAEL
WILSON

COST CENTER: 123                                          DEPARTMENT: 4

EMPLOYMENT DATE: 082596                                   STATUS: E

SALARY: 1234

ADDRESS: 7250 MICHIGAN                                    PHONE: (214)
555-1414

CITY/STATE/ZIP: PARIS, TEXAS 73000                        BIRTH DATE:
051067

F02=HELP F05=MAINMENU F07=ADDEMP2

TIRM000E: SYSTEM ERROR OCCURRED - CONTACT SUPPORT
```

Customizing the Exit

Copy Member TIRTERMA from the CA Gen CEHBSAMP library to one of your libraries.

Unlike the other user exits, the skeleton exit is not the source for the default exit in the load library. In prior releases of CA Gen, the termination exit was written in Assembler Language and named TIRTERM. The default exit TIRTERMA in the load library calls TIRTERM for compatibility with prior releases.

When you have completed your modifications, install the exit.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRMTQB—Runtime Message Table Exit

The Runtime Message Table Exit is called whenever a runtime error message is to be displayed. It contains a table of the default CA Gen runtime error messages.

Source Code

The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRMTQB is as follows:

```
01  RUNTIME-PARM1                PIC X.
01  RUNTIME-PARM2                PIC X.
01  MSG-TABLE-NAME               PIC X(8).
01  MSG-NUMBER                   PIC S9(4) COMP.
01  RETURN-MSG.
     03  RETURN-MSG-LENGTH        PIC S9(4) COMP.
     03  RETURN-MSG-ID.
         05  FILLER               PIC X(4).
         05  RETURN-MSG-NUM       PIC 999.
         05  FILLER               PIC X(4).
     03  RETURN-MSG-TEXT          PIC X(245).
     01 GLOBDATA                          size 3645 bytes.
```

Purpose

This message table exit is called by the runtime when a system-level message is to be displayed. The user can customize the wording of the messages within this exit. Additional tables can also be defined to support other dialects.

The default table includes an entry for each CA Gen runtime error message. Each entry includes the following information:

- **Message Number**—The message number is permanently assigned by CA Gen. Each message has a unique number.
- **Message Text**—The message text is the actual words that appear on the application screen when an error occurs. The message text, and any variable values that can be appended, is truncated if it exceeds the length of the error message line defined for the application screen. The error message line is a maximum of 80 characters of which 12 are reserved for the message number.

If the message number is not in the table, TIRMTQB returns a default message.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	IO-PCB
RUNTIME-PARM2	input	ALT-IO-PCB
MSG-TABLE-NAME		
MSG-NUMBER		
RETURN-MSG-LENGTH		
FILLER		
RETURN-MSG-NUM		
RETURN-MSG-TEXT		
GLOBDATA	input	Global data, used internally.

Return Code

TBD

Default Processing

The table in the default exit is used to retrieve runtime error message text.

Customizing the Exit

Copy the default exit from the CA Gen CEHBSAMP library to one of your libraries. The member name is TIRMTQB2. The text of the messages can be customized the way you want. Additional tables can be coded to support runtime error messages in other dialects for multilingual applications. The table name to be used for a given dialect is specified using the System Defaults option of the Design toolset.

When you have completed your modifications, install the exit.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRIDTRX—IMS Server Debug LTERM

IMS Server Debug LTERM Information User Exit

Source Code

TBD

Purpose

Obtain information about the LTERM to be used for trace data produced by the new dynamic runtime module TIRIRUNC.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
IO-PCB	Input/ Output	IO-PCB
ALT-IO-PCB	Input/ Output	ALT-IO-PCB
TIRTMXSZ	Input/ Output	max hex bytes of trace data that are stored (display/printed)
TIRTLTRM	Input/ Output	LTERM where trace data is sent to
TIRTMODN	Input/ Output	MFS MODNAME of LTERM

Return Code

TIRTMXSZ exit returns zero; TIRTLTRM returns spaces; and TIRTMODN returns MFS mode name DFS.EDT.

Default Processing

TBD

Customizing the Exit

TBD

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRIEX—Enhanced Map Input Edit Exit

TIRIEX is provided so that the user can modify the standard CA Gen input editing function for the enhanced map generation mode.

Source Code

This exit is used by Enhanced Map Screens only. The source code for this exit is in CA Gen CEHBSAMP library, in member TIRIEX. The sample exit provided is written in Assembler and uses standard OS Linkage.

The Linkage Parameter list used by TIRIEX is as follows:

```
ARG_RT1            DS    A            I/O PCB OR EIB
ARG_RT2            DS    A            ALT I/O PCB OR COMMAREA
ARG_IEX_COMMAREA   DS    A            PTR TO IEX COMM AREA
ARG_PATTR_DESC     DS    A            PTR TO ATTRIBUTE DESCRIPTOR
ARG_PFIELD_DESC    DS    A            PTR TO FIELD DESCRIPTOR
ARG_PIMAGE_DESC    DS    A            PTR TO IMAGE DESCRIPTOR
ARG_PEP_DESC       DS    A            PTR TO EDIT PATTERN DESC
ARG_PWORK          DS    A            PTR TO 4K WORK AREA
ARGGDTA            DS    A            GLOBDATA
```

Purpose

TIRIEX is provided so that the user can modify the standard Gen input editing function for the enhanced map generation mode. The following types of data are inputs for this exit:

- Date
- Time
- Time Stamp
- Numeric Data
- Text
- Picture (Numeric Text)

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
ARG_RT1	input/ output	IO-PCB
ARG_RT2	input/ output	ALT-IO-PCB
ARG_IEX_COMMAREA	input/ output	Address of the exit control block
ARG_PATTR_DESC	input/ output	Address of attribute descriptor
ARG_PFIELD_DESC	input/ output	Address of the field descriptor
ARG_PIMAGE_DESC	input/ output	Address of the image descriptor
ARG_PEP_DESC	input/ output	Address of the edit pattern descriptor
ARG_PWORK	input/ output	Reentrant work area
ARGGDTA	input/ output	Address of Globdata

Return Code

No explicit return code is set by the user exit.

Default Processing

The Default Input Edit Exit does not perform any processing.

Customizing the Exit

Modify the exit to perform the specific desired functions using the instructions in the exit source code file.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

TIRIEXS – Standard Map Input Edit Exit

The TIRIEXS exit is called by any blockmode application containing Standard Map screens.

Source Code

The sample source for this exit can be found in member TIRIEXS in the CA Gen CEHBSAMP library. The sample exit is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                PIC X.

01  RUNTIME-PARM2                PIC X.

01  TIRIEXS-CMCB.

   03  TIRIEXS-RETURN-CODE          PIC X.

      88  RC-ACCEPT                 VALUE '0'.

      88  RC-REJECT                 VALUE '1'.

      88  RC-REPROCESS              VALUE '2'.

      88  RC-EXIT-VALUE             VALUE '3'.

      88  RC-ERASE                  VALUE '4'.

   03  TIRIEXS-STATUS-CODE1         PIC X.

      88  SC-OK                     VALUE ' '.

      88  SC-FAIL-PENDING           VALUE 'F'.

   03  TIRIEXS-STATUS-CODE2         PIC X.

      88  SC-FIRST-PASS             VALUE '1'.

      88  SC-REENTER                VALUE '2'.

   03  TIRIEXS-ERROR-MSG-NUMBER     PIC S9(4) COMP

   03  INPUT-VALUE                  PIC X(256).

   03  INPUT-VALUE-CHAR

         REDEFINES INPUT-VALUE

         OCCURS 256 TIMES          PIC X.

   03  FIELD-LENGTH                 PIC 9(4).
```

```
03  FIELD-FILL-CHAR              PIC X(2).
03  FIELD-BEGIN-ROWCOL           PIC 9(8).
03  FIELD-END-ROWCOL             PIC 9(8).
03  ATTRIBUTE-VALUE              PIC X(256).
03  ATTRIBUTE-VALUE-CHAR
      REDEFINES ATTRIBUTE-VALUE
      OCCURS 256 TIMES          PIC X.
03  ATTRIBUTE-LENGTH             PIC 9(4).
03  ATTRIBUTE-TYPE               PIC X.
  88  ATTR-TEXT                  VALUE 'T'.
  88  ATTR-VARCHAR               VALUE 'V'.
  88  ATTR-NUMERIC               VALUE 'N'.
03  ATTRIBUTE-DECIMAL-PLACES     PIC 9(2).
03  ATTRIBUTE-CASE-SENSITIVE     PIC X.
03  MAPNAME                      PIC X(8).
03  MODNAME                      PIC X(8).
03  DIALECT-NAME                 PIC X(8).
03  DECIMAL-INDICATOR            PIC X.
03  THOUSANDS-INDICATOR          PIC X.
03  CURRENCY-INDICATOR           PIC X.
03  TXT-ORIENTATION              PIC X.
03  NUM-ORIENTATION              PIC X.
03  EDIT-PATTERN-CLASS           PIC X.
  88  EPAT-ALPHANUMERIC          VALUE 'T'.
  88  EPAT-NUMERIC               VALUE 'N'.
```

```
        88  EPAT-DATE              VALUE 'D'.
        88  EPAT-TIME              VALUE 'M'.
        88  EPAT-TIMESTAMP         VALUE 'Q'.
        88  EPAT-NONE              VALUE ' '.
    03  FILLER                 PIC X(100).
01          GLOBDATA                size 3645 bytes.
```

Purpose

TIRIEXS is provided to allow customization of the input editing behavior for Standard Map screens. This exit is used by any blockmode application containing Standard Map screens and is called for each input screen field.

Arguments

The following table gives a brief description of each of the arguments.

Name	Input/ Output	Description
RUNTIME-PARM1	input	DFHEIBLK (CICS) I/O PCB (IMS) Emulated I/O PCB (TSO)
RUNTIME-PARM2	input	DFHCOMMAREA (CICS) Alternate I/O PCB (IMS) Emulated Alt I/O PCB (TSO)
TIRIEXS-CMCB	input/output	A structure containing the following items:

Name	Input/ Output	Description
TIRIEXS-RETURN-CODE	output	The return value indicating what action the exit wants the runtimes to take for the current screen field. RC-ACCEPT - The exit took no action for the current screen field and accepts the results of Gen's validation. Normal processing will continue. RC-REJECT - The exit requests that the value input in the current screen field be marked in error. This return code should only be used for screen fields that have an edit pattern defined or for screen fields mapped to view attributes that are mandatory or that have permitted values defined. RC-REPROCESS - The exit modified the value that was input in the current screen field (INPUT-VALUE) and requests that this modified input value be revalidated. RC-EXIT-VALUE - The exit modified the value that was stored in the view attribute mapped to the current screen field (ATTRIBUTE-VALUE) and requests that this modified attribute value replace the value that was determined by Gen. This modified attribute value will not be revalidated. RC-ERASE - The exit requests that the value input in the current screen field be erased. This return code should not be used for screen fields mapped to mandatory view attributes.

Name	Input/ Output	Description
TIRIEXS-STATUS-CODE1	input	The status of the validation performed by Gen for the current screen field. SC-OK - The current screen field's value is considered to be valid by Gen's validation routines. SC-FAIL-PENDING - The current screen field's value is considered to be invalid by Gen's validation routines.
TIRIEXS-STATUS-CODE2	input	The processing status of the current screen field. SC-FIRST-PASS - This indicates this is the first time the exit has been called for the current screen field. SC-REENTER - This indicates the exit has been called previously for the current screen field and the exit requested that the field be reprocessed (RC-REPROCESS).
TIRIEXS-ERROR-MSG-NUMBER	input	The error message number determined by Gen's validation routines prior to calling the exit.
INPUT-VALUE	input/output	The value entered in the current screen field. This value should be modified by the exit if the exit returns RC-REPROCESS.
FIELD-LENGTH	input	The length of the current screen field.
FIELD-FILL-CHAR	input	The fill character defined for current screen field.
FIELD-BEGIN-ROWCOL	input	The beginning row and column of the current screen field.
FIELD-END-ROWCOL	input	The ending row and column of the current screen field.
ATTRIBUTE-VALUE	input/output	The value to be stored in the view attribute mapped to the current screen field. This value should be modified by the exit if the exit returns RC-EXIT-VALUE.

Name	Input/ Output	Description
ATTRIBUTE-LENGTH	input	The length of the view attribute mapped to the current screen field.
ATTRIBUTE-TYPE	input	The datatype of the view attribute mapped to the current screen field. ATTR-TEXT - A fixed-length text attribute. ATTR-VARCHAR - A varying-length text attribute. ATTR-NUMERIC - A numeric attribute.
ATTRIBUTE-DECIMAL-PLACES	input	The number of decimal places defined for the view attribute mapped to the current screen field.
ATTRIBUTE-CASE-SENSITIVE	input	The case sensitivity property defined for the view attribute mapped to the current screen field.
MAPNAME	input	The name of the current screen.
MODNAME	input	The modname of the current screen.
DIALECT-NAME	input	The name of the dialect used by the current screen.
DECIMAL-INDICATOR	input	The character used to represent the decimal place in the current dialect.
THOUSANDS-INDICATOR	input	The character used to represent the thousands separator in the current dialect.
CURRENCY-INDICATOR	input	The character used to represent the currency symbol in the current dialect.
TXT-ORIENTATION	input	The orientation of text fields in the current dialect.
NUM-ORIENTATION	input	The orientation of numeric fields in the current dialect.

Name	Input/ Output	Description
EDIT-PATTERN-CLASS	input	The edit pattern class for the current screen field. EPAT-ALPHANUMERIC - An alphanumeric edit pattern. EPAT-NUMERIC - A numeric edit pattern. EPAT-DATE - A date edit pattern. EPAT-TIME - A time edit pattern. EPAT-TIMESTAMP - A timestamp edit pattern. EPAT-NONE - No edit pattern is defined for the current screen field.
FILLER	input	Filler, for future use
GLOBDATA	input	Global data, used internally

Return Code

Update TIRIEXS-RETURN-CODE with the relevant value.

Default Processing

The default processing of this exit is to take no action for the current screen field and return RC-ACCEPT in TIRIEXS-RETURN-CODE.

Customizing the Exit

Copy the TIRIEXS exit to one of your libraries and modify the exit to perform the desired input editing behavior.

Building on z/OS

For information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

z/OS Blockmode User Exits—TSO

TIRTIARX—DB2 Message Exit

z/OS Dialog Managers use the CA Gen DB2 Message Exit.

Source Code

The DB2 Message Exit is used by all applications targeting DB2 database on z/OS. The source code for this exit's TSO application is in CA Gen CEHBSAMP library, in member TIRITIAX. The sample exit provided is written in COBOL and uses standard OS Linkage.

LINKAGE SECTION

```
01  RUNTIME-PARM1                PIC X.
01  RUNTIME-PARM2                PIC X.
01  TIRFAIL-SQLCA                PIC X.
01  TIRTIAR-ERRORS               PIC X.
01  TIRTIAR-TEXT-LEN             PIC X.
01  TIRTIAR-WORKAREA                    PIC X.
01 GLOBDATA                            size 3645 bytes.
```

Purpose

The TIRFAIL subroutine of the Dialog Manager calls the DB2 Message exit, TIRTIARX, whenever an unrecoverable DB2 failure occurs. TIRTIARX then calls the subroutine DSNTIAR to convert the SQL code into text. The messages returned by DSNTIAR are then merged with the runtime error messages.

TIRTIARX exit must be a DLL in order to be invoked by Gen applications, even by those using Compatibility option. DSNTIAR and DSNTIAC are provided by IBM as non-DLL modules. Therefore they need to be invoked by via TIRLGLOD.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	Emulated IO-PCB
RUNTIME-PARM2	input	Emulated ALT-IO-PCB
TIRFAIL-SQLCA	input	SQLCA
TIRTIAR-ERRORS	input	Error message lines.
TIRTIAR-TEXT-LEN	input	Length of one error message line.
TIRTIAR-WORKAREA	input	Workarea.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Behavior

As provided by CA Gen, the default exit dynamically calls DSNTIAR and is compatible with prior releases. However the sample code also contains examples of how to call DSNTIAR or DSNTIAC statically.

The call to TIRTIARX is made when TIRFAIL is building the table of messages and occurs prior to calling the default termination exit. For more information, see the Online Termination Exit and Batch Termination Exit.

Customizing the DB2 Message Exit

Copy the default exit to one of your own libraries. The member name for IMS is TIRITIAX. The default exit includes example code for the two possible combinations of calls. There are dynamic and static calls of DSNTIAR. Simply comment out the default call and remove the comments from the one you want to use.

When you have completed your modifications, install your exit.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRIRTRX—Default Retry Limit Exit

z/OS Dialog Managers for TSO and IEFAE blockmode applications use the CA Gen Default Retry Limit Exit.

Source Code

This exit is used by IMS and IEFAE blockmode applications only. The source code for this exit is in CA Gen CEHBSAMP library, in member TIRIRTRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRIRTRX is as follows:

```
01  RUNTIME-PARM1                          PIC X
01  RUNTIME-PARM2                          PIC X.
01  RETRY-TIMES                            PIC S9(4) COMP.
01  GLOBDATA                               size 3645 bytes.
```

Purpose

This exit is called at the beginning of a CA Gen IMS or IEFAE blockmode application to enable the defined default value for the TRANSACTION RETRY LIMIT system attribute to be modified.

TRANSACTION RETRY LIMIT will be initialized to this value at the beginning of each new transaction. This value may subsequently be modified by a SET TRANSACTION RETRY LIMIT statement in an action diagram. TRANSACTION RETRY LIMIT is used to specify the maximum number of times a transaction is to be retried when one of the following events occurs:

- A RETRY TRANSACTION action diagram statement executes.
- A deadlock or timeout occurs trying to access a database, and no WHEN DATABASE DEADLOCK OR TIMEOUT statement was provided for that entity action statement. (This is not applicable for z/OS transactions running under IMS. An application running under IMS that encounters a deadlock or timeout will be terminated immediately by IMS itself, even if it has a WHEN DATABASE DEADLOCK OR TIMEOUT statement provided.)

In either of these cases, any uncommitted database updates will be rolled back, and an attempt will then be made to execute the application again. Once the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, no more retries can occur, and the application will fail with a runtime error if the last retry attempt was unsuccessful.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	Emulated IO-PCB
RUNTIME-PARM2	input	Emulated ALT-IO-PCB
RETRY-TIMES	input/ output	The maximum number of times the transaction execution is retried.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

The default processing of this exit is not to modify the Default Retry Limit Exit, that is set to 10. If the Default Retry Limit Exit is used, it must not return a value greater than that specified in the Ultimate Retry Limit Exit.

Customizing the Exit

Modify the source code to set the RETRY-TIMES to the number of retries the applications should use.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRIURTX—Ultimate Retry Limit Exit

Ultimate Retry Limit Exit for TSO.

Source Code

The source code for this exits is in CA Gen CEHBSAMP library in member TIRCURTX. The sample exit is written in COBOL and uses standard OS linkage.

The Linkage Parameter list used by TIRCURTX is as follows:

```
01  RUNTIME-PARM1               PIC X.
01  RUNTIME-PARM2               PIC X.
01  ULTIMATE-RETRY-LIMIT        PIC S9(9)   COMP.
01 GLOBDATA                     size 3645 bytes
```

Purpose

The Ultimate Retry Limit Exit is used by all applications targeting DB2 database on z/OS. The Ultimate Retry Limit Exit allows the user to specify a maximum value for the TRANSACTION RETRY LIMIT system attribute. This value may never be exceeded, either by a SET TRANSACTION RETRY LIMIT statement in an action diagram, or by the Default Retry Limit Exit.

For an explanation of when and how the TRANSACTION RETRY LIMIT system attribute is used see Default Retry Limit Exit in this chapter.

This exit provides a safeguard in case the system attribute TRANSACTION RETRY LIMIT is set to an excessive value by an action diagram. Once the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, no more retries can occur, and the application will fail with a runtime error if the last retry attempt was unsuccessful.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	Input/ output	Emulated IO-PCB
RUNTIME-PARM2	Input/ output	Emulated ALT-IO-PCB
ULTIMATE-RETRY-LIMIT	Input/ output	The absolute limit which is defaulted to 99
GLOBDATA	Input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

If the Ultimate Retry Limit Exit is not modified, the maximum value of TRANSACTION RETRY LIMIT will be 99. The Ultimate Retry Limit Exit may be modified to return a value of zero to suppress all retry attempts.

Customizing the Exit

Copy TIRIURTX exit to one of your libraries or directories. For TSO and IEFAE; applies only when RETRY TRANSACTION statement executes.

Modify the copied exit as needed. When you have completed your modifications, install the exit as described in Customizing and Installing z/OS User Exits.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRSYSIX—System ID Exit

z/OS Dialog Managers use the CA Gen System Identification Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library in member TIRTSYSX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRSYSIX is as follows:

```
LINKAGE SECTION.
01  RUNTIME-PARM1        PIC X.
01  RUNTIME-PARM2        PIC X.
01  LOCAL-SYSTEM-ID      PIC X(8)
```

Purpose

The purpose of TIRSYSIX is to enable logic that lets the same application be implemented on multiple systems and perform processing specific to each system targeted.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	Emulated IO-PCB
RUNTIME-PARM2	input	Emulated ALT-IO-PCB
LOCAL-SYSTEM-ID	output	The identifier of the system where the application is executing.

Return Code

No explicit return code is set by the user exit.

Default Processing

The literal TSO is returned.

Customizing the Exit

TIRSYSIX can be modified to populate the LOCAL-SYSTEM-ID as required by the application.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRALLOX
- TIRPROUX

TIRUSRIX—User ID Exit

z/OS Dialog Managers use the CA Gen User Identification Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library in member TIRIUSRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRIUSRX is as follows:

```
01  IO-PCB.
    03  IO-PCB-LTERM                   PIC X(8).
    03  FILLER                         PIC X(2).
    03  IO-PCB-STATUS                  PIC X(2).
    03  IO-PCB-INPUT-PREF.
        05  IO-PCB-DATE                PIC S9(7) COMP SYNC.
        05  IO-PCB-TIME                PIC S9(7) COMP SYNC.
        05  IO-PCB-MSG-SEQ             PIC S9(7) COMP.
    03  IO-PCB-MAPNAME                 PIC X(8).
    03  IO-PCB-USER-ID.
        05  IO-PCB-USER-ID-C1          PIC X.
        05  FILLER                     PIC X(7).
01  ALT-IO-PCB                         PIC X.
01  FILLER-PARM                        PIC X.
01  TIRUSRID-PARM.
    05  IET-USER-ID                    PIC X(8).
    05  IET-USER-ID2                   PIC X(8).
```

Purpose

The purpose of TIRUSRIX is to obtain the userid and terminal ID of the executing application so that these values can be used as part of the key for the DB2 Profile Table and in the application itself.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description		
RUNTIME-PARM1	input	Emulated IO-PCB		
RUNTIME-PARM2	input	Emulated ALT-IO-PCB		
FILLER-PARM	input	Not used.		
TIRUSRID-PARM	output	A pointer to a structure containing the following items:		
		IET-USER-ID	output	The userid to be used in the application.
		IET-USER-ID2	output	The ID to be used as part of the Profile Table key.

Name	I/O	Description

Return Code

No explicit return code value is set by the user exit.

Default Processing

If the user ID from the IO-PCB is valid the exit returns its value for both fields otherwise the terminal ID from the IO-PCB is returned for both fields.

Customizing the Exit

Copy the TIRUSRIX to one of your libraries and modify to populate either IET-USER-ID or IET-USER-ID2 as required by the application.

Note: IET-USER-ID is used by the application as its User Identifier while IET-USER-ID2 is used as part of the Key to the RPROF (Profile Manager) Table.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

TIRSECRX

TIRSECRX—Security Interface Exit

z/OS Dialog Managers use the CA Gen Security Interface Exit.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRSECRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                          PIC X.
01  RUNTIME-PARM2                          PIC X.
01  TIRSECR-CMCB.
    03  TIRSECR-USERID                     PIC X(8).
    03  TIRSECR-TRANCODE                   PIC X(8).
    03  TIRSECR-TERMINAL-ID                PIC X(8).
    03  TIRSECR-SYSTEM-ID                  PIC X(8).
    03  TIRSECR-LOAD-MODULE                PIC X(8).
    03  TIRSECR-PSTEP-NAME                 PIC X(32).
    03  TIRSECR-DIALECT                    PIC X(32).
    03  TIRSECR-RETURN-CODE                PIC XX.
    03  TIRSECR-FAILURE-MSG                PIC X(80).
01 GLOBDATA                                size 3645 bytes.
```

Purpose

This exit is called by both cooperative and non-cooperative applications to allow transaction-level security checking to be implemented.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description	
RUNTIME-PARM1	input	Emulated IO-PCB	
RUNTIME-PARM2	input	Emulated ALT-IO-PCB	
TIRSECR-CMCB	input/ output	A structure containing the following items:	
	TIRSECR-USERID	input	The userid under which this transaction is executing, as provided by the TIRUSRIX exit.
	TIRSECR-TRANCODE	input	The load module transaction code.

Name	I/O	Description
	TIRSECR-TERMINAL-ID	input The terminal ID used by this transaction, spaces if this is a non-terminal transaction.
	TIRSECR-SYSTEM-ID	input The system ID where this transaction is executing, as provided by the TIRSYSIX exit.
	TIRSECR-LOAD-MODULE	input The load module name.
	TIRSECR-PSTEP-NAME	input The Procedure Step name.
	TIRSECR-DIALECT	input The dialect used by this application.
	TIRSECR-RETURN-CODE	output A 2-byte character field returning the result of the security check. The following values are supported: SPACES—TIRSECR-ALL-OK Anything else—failure
	TIRSECR-FAILURE-MSG	output An 80-byte character field, to be populated by this exit, to describe the failure with a message of choice.

Return Code

Update TIRSECR-RETURN-CODE with the relevant value.

Default Processing

The default processing of this exit is to do no security checking and to return TIRSECR-ALL-OK as the return code.

Customizing the Exit

Copy the TIRSECRX exit to one of your libraries and modify to perform security checking as required by the application. Ensure that TIRSECR-RETURN-CODE is set to spaces when the security check is successful or some other value to indicate failure. If a message describing the violation is returned in TIRSECR-FAILURE-MSG, the Dialog Manager will pass it to TIRTERMA.

Building on z/OS

For information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRUSRIX
- TIRSECVX
- TIRELOGX
- TIRTERMA

TIRDATX—Date and Time Services Exit

z/OS Dialog Managers use the CA Gen Date and Time Services Exit.The Date and Time Services Exit can be used to intercept, adjust, or validate system dates and times. This exit is provided to allow user modification and customization of date and time processing.

Source Code

The source code for this exit is in the Gen CEHBSAMP library in member TIRDATX. The sample exit provided is written in Assembler and uses standard OS Linkage.

The parameter list used by TIRDATX is as follows:

```
PARMRT1  DS    A
PARMRT2  DS    A
PARMCMCB DS    A
PARMWORK DS    A
PARMGDTA DS    A
```

Purpose

This exit receives control for some but not all date and time services. Only services that acquire, or manipulate the date and time, where that date or time was acquired from the system, or where validation is involved, invoke this exit.

This exit is not invoked for the following conditions:

- Services involving conversion from one form to another does not invoke this exit.
- If some error condition exists. For example, if the clock is not set, the date and time services return the error directly to the requester and do not call this exit.
- For validation, if the value is not valid, the failure is returned to the requester and the exit is not called.
- If this exit changes a date or time and requests re-validation, and the value is in error, the error is returned to the requester and the exit is not called.

Note: If the date and time is modified by the exit, the exit must indicate this by returning the appropriate return code. Return codes that are invalid (not one of the listed values) will be ignored and the result is as if the exit returned zero (0). Therefore it is imperative that you not take advantage of any behavioral aspects not explicitly documented here or in the sample code since future releases could change the operation.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
PARMRT1	input/ output	EIB or IO PCB
PARMRT2	input/ output	COMMAREA or Alt IO PCB
PARMCMCB	input/ output	Address of the Date Communication Control Block
PARMWORK	input/ output	Address of a 256 byte workarea
PARMGDTA	input/ output	Address of GLOBDATA

Return Code

Default Processing

Sample Code

Sample code has been provided as commented out blocks of code. This is as an example only. To use the sample code, you must remove the comments. Sections that are specific to a particular system, such as TSO, are indicated by comments preceding and following the code. Common code that should be used by both examples is also indicated.

The sample code provides the ability to respond to request code 1 (get system date) and request code 7 (get current timestamp). The date information is read from a file and is used to change the year, month, and day. The timestamp information (hours, minutes, seconds, microseconds) is read and left unchanged.

Delivery Configuration

As configured, the sample code that will read the required date from a file when generated for TSO. This exit can also obtain values from DB2 table lookups.

To use this facility, you must change the source code to set the variable as appropriate. For TSO mode, use the assembler but do not use the preprocessor.

Registers

Register 14 contains the address that control is to be returned to, and Register 13 contains the address of a savearea set up for the exit's usage. All other registers must be saved and restored on return.

Customizing the Exit

You can customize the exit to perform your specific needs. The following paragraphs provide guidelines to be observed when modifying this exit. Be sure to read all notes provided with the sample code for the latest information on using this exit.

DREQ Service Request

This exit uses operating system standard linkage. On return, registers 0 - 14 must be restored to their values on entry. Register 15 contains a return code to control the processing of the date and time services. The service request code is indicated in the Date CMCB field, DREQ. The return codes and service requests are discussed later in this section.

I/O Format

The format of input and output data are indicated in the CMCB fields DDATEF and DTIMEF. These values should be examined to determine the format of the data to be stored, or to be used as input by the exit.

Fields in the Date CMCB

Other fields in the CMCB have various meanings and formats as described in the following paragraphs.

DDATE

This field contains the binary date value. It is treated as a signed decimal number and converted to binary. The format is specified by the field DDATEF and cannot be changed. This field can be in one of the following formats:

- YYYYMMDD—Four digit year, two digit month, and two digit day
- YYMMDD—Two digit year (the century is omitted), two digit month, and two digit day
- CYYMMDD—One digit century code, two digit year, two digit month, and two digit day

Note: The one digit century code (C) is a number from 0 to 9, inclusive. The century ranges that can be represented are from 1600 to 2599, inclusive. The century codes are: 0 = 19XX, 2 = 20XX, 3 = 22XX, 4 = 23XX, 5 = 24XX, 6 = 25XX, 7 = 16XX, 8 = 17XX, and 9 = 18XX.

DDATEF

This field contains an indicator of the format of the DDATE field's content and cannot be changed.

DTIME

This field contains the binary time value. The time is treated as a signed decimal number with the format HHMMSSTH, or HHMMSST, or HHMMSS with the following conventions:

- HH—Hours
- MM—Minutes
- SS—Seconds
- T—Tenths of seconds
- H—Hundredths of seconds

The format used is specified by the DTIMEF field.

DTIMEF

This field contains an indicator of the format of the DTIME field's content and cannot be changed.

DTSTAMP

This field contains the zoned decimal time stamp value in a fixed format of YYYYMMDDHHMISSNNNNNN with the following conventions:

- YYYY—Four-digit year
- MM—Two-digit month
- DD—Two-digit day
- MI—Two-digit minutes
- SS—Two-digit seconds
- NNNNNN—Six-digit microseconds

DINC

This field contains a signed binary increment to be added to the date value in DDATE when DREQAS service is requested. It is unused in all other cases. A negative value will result in a date prior to the base date.

Testing the DREQ Field

The exit must test the DREQ field of the Date CMCB to determine the service request made of the Date/Time routine. This is used to customize the exit based on your needs. For example, if you wish to perform local validation of dates only, the request of interest is DREQVAL. For all other requests, the exit must return a zero.

Modifying Date and Time

If the exit is used to modify date or time, the exit must modify the appropriate fields for the service request. Different service requests use different areas of the Date CMCB as their input, and place their output in various fields.

Service Requests Intercepted by the Date and Time Services Exit

The service requests intercepted by this exit are:

- DREQVAL—Request date and/or time validation
- DREQSD—Return the current system date and time
- DREQAS—Add a specified increment to the date value
- DREQST—Return the current timestamp
- DREQVTS—Validate the timestamp provided
- DREQST—Return the current system timestamp

DREQ Return Codes

The return codes and their meanings vary for the different service requests indicated in the DREQ field. Refer to the following paragraphs for the request, return codes and meaning.

DREQ - Service Request DREQVAL

- 0—Use the system date/time as stored in DCMCB. The exit has not modified these values and accepts them as they are.
- 4—The exit has modified the date/time stored in the DCMCB and requests that the validation be re-executed for these values. Note that the exit will be called again after validation is complete.
- 8—The exit has modified the date/time stored in the DCMCB and does not require the values be re-validated. The modified values are returned to the requester.
- 12—The exit requests that the date/time service fail the request. This is returned to the service requester as if the date and/or time were invalid.
- DREQ - Service Request DREQSD, DREQAS, DREQVST, and DREQST
- 0—Use the system date/time as stored in DCMCB.
- 4—The exit requests the date, time, or timestamp value be recomputed. If the exit has modified any of these values, the modifications are discarded and the values computed from the system clock. For DREQAS, the DINC value represents the number of days to be added to the date. The exit is called again after the date and time have been recomputed.
- 8—The exit has modified the date/time stored in the DCMCB and does not require the date/time services recompute the associated values. The modified and unmodified values are returned to the service requester unchanged.
- 12—The exit requests that the date/time service fail the request. This is the same as if the clock was not set or was damaged. The following table describes service requests and the fields they use:

Service Request (DREQ)	Input	Output	Applicable Notes
DREQVAL	DDATE, DTIME		1, 4, 5
DREQAS	DDATE, DTIME, DINC	DDATE, DTIME	2, 5
DREQVTS	DREQSD	DDATE, DTIME	2, 5
DREQST		DTSTAMP, DDATE, DTIME	2, 3, 5

Note:

1. Date and/or Time validation can be skipped if the appropriate field is set to zero. For example, if DDATE is zero, then the Date validation is skipped.
2. Initial processing obtains the current date and time using the system clock and adjusts the value based on the time zone adjustment. If the request is DREQAS, then the DINC value is added to the number of days prior to computing the Gregorian date, and then the DDATE / DTIME fields are computed. If the exit requests that the values be reprocessed, any modification that the exit made to the DDATE / DTIME fields is discarded and the values recomputed from the system clock. DINC can be altered if the request was DREQAS.
3. If the request is DREQST, then the system time stamp values are computed from the clock values.
4. Validation returns a code to the requester indicating the validity of the date/time/time stamp. If the value is valid, the exit is called or recalled if the exit requested the validation be reprocessed.
5. The formats of input and output data are indicated in the CMCB fields DDATEF and DTIMEF. These values should be examined to determine the format of the data to be stored, or to be used as input by the exit.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None

TIRDEVT—Device Characteristics Exit

Device Characteristics exit for TSO

Source Code

TIRDEVT exit is written in IBM Macro Assembly so it can use system service calls that are not available through COBOL.

On entry, register 1 addresses an OS-standard argument list, consisting of five arguments. These are as follows:

1. Address of emulated IO-PCB.
2. Address of emulated ALT-IO-PCB.

3. Address of the device capabilities information data area to be modified by this exit and returned to the caller.
4. The address of an 8-byte field containing the name of the current dialect.
5. GLOBDATA

Purpose

The Device Characteristics exit, TIRDEV, is called every time a message is sent from or received by an application. This exit provides the runtime data stream processing routines for the definition of the specific device capabilities.

Arguments

Five pointer parameters are passed when this routine is called. The parameters for TSO are defined in the following table.

Name	I/O	Description
Parm 1	Input/output	emulated IO-PCB
Parm 2	Input/output	emulated ALT-IO-PCB
Parm 3	Input/output	Return characteristic
Parm 4	Input/output	TMOHDLCT (dialect)
Parm 5	Input/output	GLOBDATA

TMOHDLCT is a pointer to the eight-character dialect name as returned from the User Dialect exit, TIRDLCTX. This value represents the current selected dialect. The default exit returns a default dialect value for this parameter.

The RETURN CHARACTERISTIC is a pointer to a 256-byte structure defined by CA Gen for return of the device capabilities

Return Code

The RETURN CHARACTERISTIC structure returns the device capabilities. The fields in this structure are as follows:

EXTPARM		Returned Device Capabilities
MAXROWR	DS H	(24/32/43/27) maximum number of screen rows
MAXCOLR	DS H	(80/132) maximum number of screen columns
EXTDSR	DS CL1	(0/255) 0= No Extended Data Stream support

EXTPARM		Returned Device Capabilities
EXTCLRR	DS CL1	(0/255) 0= Base Color, 255 = Extended Color
EXTHIGHR	DS CL1	(0/255) 0= No Highlight, 255 = Highlighting
EXTGRID	DS CL1	(0/255) 0= No Grid Line, 255 = Grid Line
EXTDBCS	DS CL1	(0/255) 0= No DBCS DISPLAY or ENTRY
EXTSCS8	DS CL1	(0/255) 0= No DBCS Set F8, SCS'8' for DBCS
XMIXENT	DS CL1	(0/255) 0= No Mixed (SBCS/DBCS) entry
XINEDIT	DS CL1	(0/255) 0= No INPUT EDITING ATTRIBUTE support
XOUTXLAT	DS A	Pointer to 256 byte Output Translate Table
XINPXLAT	DS A	Pointer to 256 byte Input Translate Table
	DS CL235	Filler MUST BE ZERO

Default Processing

The maximum row and column values are derived from the 3270 model type. At this time, CA Gen supports only IBM 3270 model 2 (24 x 80).

Extended Data Stream support and other extended attribute capabilities of the terminal are derived from query or any other user defined method of retrieving the terminal status. If Extended data stream is not enabled, then no extended data stream functions are built into the outbound data stream. If Double Byte Character Support (DBCS) is not enabled, then no DBCS data is placed in the outbound data stream. If MIXENT is not enabled, all mixed entry fields are built as Single Byte Character Support (SBCS) only fields in the data stream.

Additional information is available in the vendor documentation on National Language Support (NLS).

Translate Tables

The output (OUTXLAT) and input (INPXLAT) tables are standard 256 byte translate tables in a format suitable for the translate (TR) operations code (op code). OUTXLAT is used when the current device does not support the same code page as the application and encyclopedia. This means that a difference exists in the code points for the encyclopedia and application database and the code points for the device. The translate table needs to convert the code points in the output data stream to the correct code points for the current device to display the correct glyphs. INPXLAT is used when data is received from the terminal to convert the code points back to the appropriate values for the application database and encyclopedia.

If the device supports the same code page as the application and database, then OUTXLAT and INPXLAT should be set to ZERO (0) to suppress any code point conversion.

For example, if the current device is a Japanese 557x terminal supporting code page 930 (uppercase Roman only) and the application prompts contain lower case Roman letters, the translate tables must perform inbound and outbound translations.

Outbound, the translate table performs monocasing (from lowercase to uppercase), and translates the application database code points to the device code points. This displays the correct glyphs on the device.

Inbound, the translate table translates the device code points to the application database and encyclopedia code points for proper storage. This prevents corruption of the data in the database.

To accomplish the translation process in the preceding example, set OUTXLAT to point to a table that converts lowercase code points to uppercase. Set INPXLAT to a table that translates device Katakana back into the code point values needed in the application database.

Customizing the Exit

Copy the default exit from the CA Gen sample library to a separate library. The member name is TIRDEVT. You can customize this exit to accept input from the User Dialect exit (TIRDLCTX) to change the code page during production.

When you have completed your modifications, install the exit as described in the section on customizing user exits.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRDLCTX—User Dialect Exit

z/OS Dialog Managers use the CA Gen User Dialect Exit.

Source Code

The source code for this exit is in the CA Gen CEHBSAMP library member TIRDLCTX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                                PIC X.
01  RUNTIME-PARM2                                PIC X.
01  TIRDLCT-CMCB.
    03  TIRDLCT-USERID                          PIC X(8).
    03  TIRDLCT-TERMINAL-ID                     PIC X(8).
    03  TIRDLCT-SYSTEM-ID                       PIC X(8).
    03  TIRDLCT-RETURN-DIALECT                   PIC X(8).
01  GLOBDATA                                    size 3645 bytes.
```

Purpose

This exit is used by all applications. The purpose of the TIRDLCTX exit is to supply the current user's dialect to the application. It is meaningful for multilingual applications.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.
TIRDLCT-CMCB	input/ output	A structure containing the following items:

Name	I/O	Description	
	TIRDLCT-USERID	input	The userid under which this transaction is executing, as provided by the TIRUSRIX exit.
	TIRDLCT-TERMINAL-ID	input	The terminal ID used by this transaction, spaces if this is a non-terminal transaction.
	TIRDLCT-SYSTEM-ID	input	The system ID where this transaction is executing, as provided by the TIRSYSIX exit.
	TIRDLCT-RETURN-DIALECT	input	The dialect used by this application.

Return Code

No explicit return code is set by the user exit.

Default Processing

The default processing of this exit is to return a dialect name of DEFAULT.

Customizing the Exit

Copy the default exit from the CA Gen CEHBSAMP library to one of your libraries. The member name is TIRDLCTX. For multilingual support, modify this module to return the appropriate dialect for a user. The dialect returned is the one selected using the Dialect Definition option of the Design Toolset. If none is selected or returned, the default dialect is used.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

TIRDEVC

TIRUPPRX—Uppercase Translation Exit

z/OS Dialog Managers use the CA Gen Uppercase Translation Exit. This exit is also called the Lower-to-Uppercase Conversion Exit.

Source Code

This exit is used by single byte and double byte applications. When used by double byte applications an alternate entry point TIRUPDBx is used. The source code for this for this exit is in CA Gen CEHBSAMP library in member TIRUPPRX. The sample exit is written in COBOL and uses OS linkage.

The Parameter list used by TIRUPPRX is as follows:

```
01  RUNTIME-PARM1                          PIC X.
01  RUNTIME-PARM2                          PIC X.
01  XLATE-TABLE-NAME                      PIC X(8).
01  XLATE-LEN                             PIC S9(4) COMP.
01  XLATE-DATA                            PIC X(4096).
01  GLOBDATA                               size 3645 bytes.
```

Purpose

The purpose of the Uppercase Translation User Exit is to translate character input from lowercase to uppercase. It contains a table of paired lower and uppercase characters. This exit is called by the Dialog Manager to translate the lower caser trancode to upper case, by the TIRFUPPR Function to translate the designated data to upper case and by the Standard Map runtime to translate the identified input data to upper case.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.
XLATE-TABLE-NAME	input	Name of the translation table to be used.
XLATE-LEN	input	Length of data to be translated.
XLATE-DATA	input/output	Data to be translated.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code value is defined for this exit.

Default Processing

The default processing of this exit is to convert lower case characters to upper case using a table named DEFAULT that contains the English character set(A-Z).

Customizing the Exit

Copy the default exit from the CA Gen CEHBSAMP library to one of your own libraries. The member name is TIRUPPRX.

The exit supports both single byte and double byte languages. Adding support for DBCS is done in the same way as for single byte.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRYYX—Two-Digit Year Input Edit Exit

z/OS Dialog Managers use CA Gen Two-Digit Year Input Exit.

Source Code

The source code for this exit is in the CA Gen CEHBSAMP library member TIRYYX. The sample exit provided is written in Assembler and uses standard OS Linkage.

The parameter list used by TIRYYX is as follows:

```
EXTCB    DS    A
WORKAREA DS    A
GLOBDATA DS    A
```

Purpose

This exit is used by CA Gen Standard Map applications only. The purpose of the TIRYYX exit is to process two-digit or YY-style date input and set the century part using any chosen algorithm to implement logic to handle the century part of the date.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
EXITCB	input	Address of the Date Communication Control Block
WORKAREA	input	Address of a 256 byte workarea
GLOBDATA	input	Address of GLOBDATA

Return Code

Before returning this exit must restore registers 0-14 to their values on entry and update register 15 with a value of 4 to indicate that the YY modified by the exit should be used. Any other value, including 0, indicates the original values passed to the exit are acceptable to continue processing.

Default Processing

The exit contains sample code for 2 algorithms but neither are executed. By default the exit returns a value of 0, indicating that no changes were done by the exit.

Customizing the Exit

Copy the TIRYYX exit to one of your libraries.

Internally, CA Gen handles four-digit year dates correctly assuming the user application uses a YYYY edit pattern throughout. If the user interface is designed to accept a two-digit date entry, and defaulting to the current century is not acceptable, use this exit to implement logic to get the required behavior for defaulting the century part of the date. The exit is called to process either a DATE or TIMESTAMP field which utilizes a 2-digit year (YY) in the edit pattern associated with the field. An indicator is set in the exit control block to indicate if the value is a date or timestamp.

Modify the exit to use one of the provided algorithms or add your own as required by your applications.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRMTQB—Runtime Message Table Exit

The Runtime Message Table Exit is called whenever a runtime error message is to be displayed. It contains a table of the default CA Gen runtime error messages.

Source Code

The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRMTQB is as follows:

```
01  RUNTIME-PARM1               PIC X.
01  RUNTIME-PARM2               PIC X.
01  MSG-TABLE-NAME              PIC X(8).
01  MSG-NUMBER                  PIC S9(4) COMP.
01  RETURN-MSG.
     03  RETURN-MSG-LENGTH       PIC S9(4) COMP.
     03  RETURN-MSG-ID.
         05  FILLER              PIC X(4).
         05  RETURN-MSG-NUM      PIC 999.
         05  FILLER              PIC X(4).
     03  RETURN-MSG-TEXT         PIC X(245).
     01 GLOBDATA                        size 3645 bytes.
```

Purpose

This message table exit is called by the runtime when a system-level message is to be displayed. The user can customize the wording of the messages within this exit. Additional tables can also be defined to support other dialects.

The default table includes an entry for each CA Gen runtime error message. Each entry includes the following information:

- **Message Number**—The message number is permanently assigned by CA Gen. Each message has a unique number.

- **Message Text**—The message text is the actual words that appear on the application screen when an error occurs. The message text, and any variable values that can be appended, is truncated if it exceeds the length of the error message line defined for the application screen. The error message line is a maximum of 80 characters of which 12 are reserved for the message number.

If the message number is not in the table, TIRMTQB returns a default message.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	Emulated IO-PCB
RUNTIME-PARM2	input	Emulated ALT-IO-PCB
MSG-TABLE-NAME		
MSG-NUMBER		
RETURN-MSG-LENGTH		
FILLER		
RETURN-MSG-NUM		
RETURN-MSG-TEXT		
GLOBDATA	input	Global data, used internally.

Return Code

TBD

Default Processing

The table in the default exit is used to retrieve runtime error message text.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRTERMA—Termination Exit

The CA Gen Termination Exit is called by z/OS Dialog Managers when a fatal runtime error is encountered.

Source Code

TIRTERMA Termination Exit is used by all non-cooperative applications targeting z/OS. The source code is in CA Gen CEHBSAMP library, in member TIRTERMA. The sample exit provided is written in COBOL and uses standard OS Linkage.

The parameters passed between the fail routine and the termination exits are defined via structure TERM-EXIT-PARM-LIST. This structure is included via copy member CBLTERM, which is also in the CEHBSAMP library.

The Linkage Parameter list used by TIRTERMA is as follows:

```
01  RUNTIME-PARM1              PIC X.
01  RUNTIME-PARM2              PIC X.
01  TERM-EXIT-PARM-LIST        structure defined in CBLTERM.
01  GLOBDATA                   size 3645 bytes.
```

Purpose

Runtime errors are handled by the Dialog Manager.

Runtime errors are either fatal or non-fatal errors. When a non-fatal error occurs, such as invalid user input, the Dialog Manager displays an error message on the transaction screen. You can correct the error and continue processing the transaction.

When a fatal error occurs, transaction processing is terminated. The Dialog Manager executes a fail routine that backs out changes by performing the necessary rollbacks of the databases. The fail routine then calls a termination exit that determines what diagnostic (error) information is displayed and where it is displayed.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input/ output	This is the DFHEIBLK which is automatically included if translated.
RUNTIME-PARM2	input/ output	This is the DFHCOMMAREA which is automatically included if translated.

Name	I/O	Description
TERM-EXIT-PARM-LIST	input/ output	Structure of parameters for termination and failure routine. The items in this structure are described in the below section called CBLTERM Field Definitions.
GLOBDATA	input	Global data, used internally.

CBLTERM Field Definitions

The structure TERM-EXIT-PARM-LIST is defined via in copy member CBLTERM. The items in this structure are as follows:

TERM-STATUS-CODE

When TIRFAIL calls TIRTERM(A/B), TERM-STATUS-CODE is used to control what TIRFAIL does next.

The following table provides a description of each TERM-STATUS-CODE value:

Value	Description
’ ’ (space) or 0 (zero)	TIRFAIL displays the message and redisplays the previous screen with TERM-DEFAULT-MSG in the error message field.
1	This value indicates that TIRTIRM(A/B) has handled the messages and will not display them. It will, however, redisplay the previous screen with TERM-DEFAULT-MSG in the error message field.
2	Thisvalue indicates that TIRTIRM(A/B) has handled everything. TIRFAIL does not display the messages and does not redisplay the previous screen.

TERM-FAIL-TYPE

The following table contains a description of TERM-FAIL-TYPE errors:

Error	Value	Decription
TERM-FAIL-DB2	P	A DB2 error occurred while accessing the RPROF (profile) table.
TERM-FAIL-IEC	I	An AllFusion Gen error occurred in the Dialog Manager.
TERM-FAIL-EXEC	E	A database error occurred in an action block or procedure.

Error	Value	Decription
TERM-FAIL-DIALOG	D	A non-database error occurred in the Dialog Manager.
TERM-FAIL-TSQ	Q	An error occurred while accessing the CICS temporary storage queue profile table.

Remaining Fields

The remaining CBLTERM fields are described in the table:

Field	Description
TERM-ERROR-ACTIONNAME	Contains the name of the action block.
TERM-DEFAULT-MSG	This is an output field that by default contains the following message: TIRM000E: SYSTEM ERROR OCCURRED - CONTACT SUPPORT The message can be changed in the termination exit to anything meaningful to the user. For online procedures with a screen, the message is visible in the error message field when the screen is redisplayed.
TERM-SYSTEM-PRINTER	TERM-SYSTEM-PRINTER is valued with the printer TERMID if the action block executed a PRINTER TERMINAL IS statement.
TERM-ERRORENCOUNTERED-SW	Indicates the message: TIRM037E: ** A FATAL ERROR HAS BEEN ENCOUNTERED **
TERM-VIEW-OVERFLOW-SW	Indicates the message: TIRM037E: ** FATAL VIEW OVERFLOW HAS BEEN ENCOUNTERED **
TERM-ACTION-ID	Is appended to the message: TIRM034E: LAST OR CURRENT DATABASE STATEMENT = ...
TERM-ATTRIBUTE-ID	Is appended to the message: TIRM040E: PERMITTED VALUES MISMATCH, FIELD = F ...
TERM-STATUS-FLAG	Produces the message: TIRM038E: ** FATAL DATABASE ERROR HAS BEEN ENCOUNTERED **
TERM-LAST-STATUS	Is appended to the message: TIRM039E: DB LAST STATUS = ...

Field	Description
TERM-TRACE-PTR	This field is documented in online help under the error message TIRM039E.
TERM-LAST-STATEMENTNUM	Is appended to the message: TIRM035E: CURRENT STATEMENT BEING PROCESSED = ...
TERM-CURR-AB-ID	Is appended to the message: TIRM032E: LAST OR CURRENT ACTION BLOCK ID = ...
TERM-CURR-AB-NAME	Is appended to the message: TIRM033E: LAST OR CURRENT ACTION BLOCK NAME = ...
TERM-EABPCB-CNT, TERM-EABPCB-ENTRY, TERM-EABPCB-PTR	These fields describe PCB pointers. The first is the IO-PCB, the second is the ALTERNATE-IOPCB; the last is a database pointer.
TERM-SQLCA-PTR	The following is a pointer to the SQLCA. The address fields of the SQLCA, first define it in linkage. Use the following example: MY-SQLCA FILLER MY-SQL-CODE FILLER Then add a SET statement at the beginning of the procedure division as shown : SET ADDRESS OF MY-SQLCA TO TERM-SQLCA-PTR
TERM-IEF-COMMAND	The special field of COMMAND.
TERM-IEF-TRANCODE	The special field of TRANCODE.
TERM-EXIT-STATE	The exit state number.
TERM-EXIT-INFOMSG	The exit state message.
TERM-USER-ID	The special field of USERID.
TERM-TRMINAL-ID	The special field of TERMID.
TERM-PRINTER-ID	Represents the ID of the system printer.
TERM-DIALOG-MESSAGENUM	The message number is the FAIL-MSG-NO set be the Dialog Manager. See the Messages Guide for the message represented by the error code displayed.

Field	Description
TERM-OUTPUT-MESSAGE	Prior to calling the termination exit, TIRFAIL, prepares a table of messages that it will display on return from the exit if the TERM-STATUSCODE is a space or a zero. These messages are available to the exit. The last line with a message is followed by a line of all spaces.
TERM-DIALECT-NAME	The current dialect
TERM-FAILURE-MESSAGETEXT	The text of the failure message. This may be moved to TERM-DEFAULT-MSG if you want it displayed on the application screen instead of the message: TIRM000E: SYSTEM ERROR OCCURRED - CONTACT SUPPORT

Return Code

TIRTERMA can return three valid status codes to the fail routine. They are used to control what screens will be displayed to the user. For example, if you want to replace the CA Gen error screen with your own and then return to the application screen, and you can modify the code to return the application screen, you can modify the code to return status code 1.

The following table provides a description:

Status Code	Error Message Screen Displayed	Application Screen Displayed
1	No	Yes
2	No	No
0, blank, other (except 1 or 2)	Yes	Yes

The skeleton exit contains example code for each of these status codes with the code for ‘1’ and ‘2’ commented out.

Default Processing

If a runtime error occurs and the default termination exit is used, processing is as follows:

1. The Dialog Manager performs all necessary rollbacks. This is done regardless of the termination exit used.
2. The Dialog Manager fail routine calls the default termination exit. It returns to the fail routine without doing anything, which causes the default termination logic in the fail routine to be used.

3. The CA Gen fail routine displays an error screen that lists the appropriate CA Gen runtime error messages. See the following error message screen:

```
TIRM030E: APPLICATION FAILED ** UPDATES HAVE BEEN BACKED OUT

TIRM031E: FAILING PROCEDURE EXIT DATA FOLLOWS

TIRM032E: LAST OR CURRENT ACTION BLOCK ID = 507774696

TIRM033E: LAST OR CURRENT ACTION BLOCK NAME = ABADDEMP

TIRM034E: LAST OR CURRENT DATABASE STATEMENT =

TIRM035E: CURRENT STATEMENT BEING PROCESSED = 10

TIRM037E: ** A FATAL ERROR HAS BEEN ENCOUNTERED **

TIRM046E: *** TRANSACTION PROCESSING TERMINATED

TIRM044E: *** PRESS PA2 TO CONTINUE ***
```

4. When you press PA2 (NEXT PAGE key) from the error message screen, CA Gen displays the last screen for the transaction that was being processed when the error occurred.

   If you are using the Testing Facility, the PA2 key is the ISPF NEXT PAGE key you defined on the Test Environment Panel.

5. CA Gen recovers all data in the import views at the time the error occurred. Therefore, all user input is recovered and displayed upon the screen. Screen fields that are only in the export view may or may not be populated, depending on when the error occurred.

6. An error message appears in the system error message area defined for the screen. This message is distinct from the runtime error messages displayed on the error message screen. The default error message is:

   SYSTEM ERROR OCCURRED - CONTACT SUPPORT.

   See the following illustration for an example of an application screen that is displayed after an error has occurred.

7. The transaction is terminated, but the application remains active and the user can continue with another transaction as shown in the following screen:

```
IEFSLSB                              CORPORATE MANAGEMENT

                                     EMPLOYEE MAINTENANCE

EMPLOYEE NUMBER: 123456                                   NAME: MICHAEL
WILSON

COST CENTER: 123                                          DEPARTMENT: 4

EMPLOYMENT DATE: 082596                                   STATUS: E

SALARY: 1234

ADDRESS: 7250 MICHIGAN                                    PHONE: (214)
555-1414

CITY/STATE/ZIP: PARIS, TEXAS 73000                        BIRTH DATE:
051067

F02=HELP F05=MAINMENU F07=ADDEMP2

TIRM000E: SYSTEM ERROR OCCURRED - CONTACT SUPPORT
```

Customizing the Exit

Copy Member TIRTERMA from the CA Gen CEHBSAMP library to one of your libraries.

Unlike the other user exits, the skeleton exit is not the source for the default exit in the load library. In prior releases of CA Gen, the termination exit was written in Assembler Language and named TIRTERM. The default exit TIRTERMA in the load library calls TIRTERM for compatibility with prior releases.

When you have completed your modifications, install the exit.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRIEX—Enhanced Map Input Edit Exit

TIRIEX is provided so that the user can modify the standard CA Gen input editing function for the enhanced map generation mode.

Source Code

This exit is used by Enhanced Map Screens only. The source code for this exit is in CA Gen CEHBSAMP library, in member TIRIEX. The sample exit provided is written in Assembler and uses standard OS Linkage.

The Linkage Parameter list used by TIRIEX is as follows:

```
ARG_RT1            DS    A            I/O PCB OR EIB
ARG_RT2            DS    A            ALT I/O PCB OR COMMAREA
ARG_IEX_COMMAREA   DS    A            PTR TO IEX COMM AREA
ARG_PATTR_DESC     DS    A            PTR TO ATTRIBUTE DESCRIPTOR
ARG_PFIELD_DESC    DS    A            PTR TO FIELD DESCRIPTOR
ARG_PIMAGE_DESC    DS    A            PTR TO IMAGE DESCRIPTOR
ARG_PEP_DESC       DS    A            PTR TO EDIT PATTERN DESC
ARG_PWORK          DS    A            PTR TO 4K WORK AREA
ARGGDTA            DS    A            GLOBDATA
```

Purpose

TIRIEX is provided so that the user can modify the standard Gen input editing function for the enhanced map generation mode. The following types of data are inputs for this exit:

- Date
- Time
- Time Stamp
- Numeric Data
- Text
- Picture (Numeric Text)

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
ARG_RT1	input/ output	Emulated IO-PCB
ARG_RT2	input/ output	Emulated ALT-IO-PCB

Name	I/O	Description
ARG_IEX_COMMAREA	input/ output	Address of the exit control block
ARG_PATTR_DESC	input/ output	Address of attribute descriptor
ARG_PFIELD_DESC	input/ output	Address of the field descriptor
ARG_PIMAGE_DESC	input/ output	Address of the image descriptor
ARG_PEP_DESC	input/ output	Address of the edit pattern descriptor
ARG_PWORK	input/ output	Reentrant work area
ARGGDTA	input/ output	Address of Globdata

Return Code

No explicit return code is set by the user exit.

Default Processing

The Default Input Edit Exit does not perform any processing.

Customizing the Exit

Modify the exit to perform the specific desired functions using the instructions in the exit source code file.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

TIRIEXS – Standard Map Input Edit Exit

The TIRIEXS exit is called by any blockmode application containing Standard Map screens.

Source Code

The sample source for this exit can be found in member TIRIEXS in the CA Gen CEHBSAMP library. The sample exit is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                PIC X.

01  RUNTIME-PARM2                PIC X.

01  TIRIEXS-CMCB.

   03  TIRIEXS-RETURN-CODE          PIC X.

      88  RC-ACCEPT                 VALUE '0'.

      88  RC-REJECT                 VALUE '1'.

      88  RC-REPROCESS              VALUE '2'.

      88  RC-EXIT-VALUE             VALUE '3'.

      88  RC-ERASE                  VALUE '4'.

   03  TIRIEXS-STATUS-CODE1         PIC X.

      88  SC-OK                     VALUE ' '.

      88  SC-FAIL-PENDING           VALUE 'F'.

   03  TIRIEXS-STATUS-CODE2         PIC X.

      88  SC-FIRST-PASS             VALUE '1'.

      88  SC-REENTER                VALUE '2'.

   03  TIRIEXS-ERROR-MSG-NUMBER     PIC S9(4) COMP

   03  INPUT-VALUE                  PIC X(256).

   03  INPUT-VALUE-CHAR

         REDEFINES INPUT-VALUE

         OCCURS 256 TIMES          PIC X.

   03  FIELD-LENGTH                 PIC 9(4).
```

```
03  FIELD-FILL-CHAR              PIC X(2).
03  FIELD-BEGIN-ROWCOL           PIC 9(8).
03  FIELD-END-ROWCOL             PIC 9(8).
03  ATTRIBUTE-VALUE              PIC X(256).
03  ATTRIBUTE-VALUE-CHAR
      REDEFINES ATTRIBUTE-VALUE
      OCCURS 256 TIMES          PIC X.
03  ATTRIBUTE-LENGTH             PIC 9(4).
03  ATTRIBUTE-TYPE               PIC X.
  88  ATTR-TEXT                  VALUE 'T'.
  88  ATTR-VARCHAR               VALUE 'V'.
  88  ATTR-NUMERIC               VALUE 'N'.
03  ATTRIBUTE-DECIMAL-PLACES     PIC 9(2).
03  ATTRIBUTE-CASE-SENSITIVE     PIC X.
03  MAPNAME                      PIC X(8).
03  MODNAME                      PIC X(8).
03  DIALECT-NAME                 PIC X(8).
03  DECIMAL-INDICATOR            PIC X.
03  THOUSANDS-INDICATOR          PIC X.
03  CURRENCY-INDICATOR           PIC X.
03  TXT-ORIENTATION              PIC X.
03  NUM-ORIENTATION              PIC X.
03  EDIT-PATTERN-CLASS           PIC X.
  88  EPAT-ALPHANUMERIC          VALUE 'T'.
  88  EPAT-NUMERIC               VALUE 'N'.
```

```
        88  EPAT-DATE              VALUE 'D'.
        88  EPAT-TIME              VALUE 'M'.
        88  EPAT-TIMESTAMP         VALUE 'Q'.
        88  EPAT-NONE              VALUE ' '.
    03  FILLER                 PIC X(100).
01          GLOBDATA                size 3645 bytes.
```

Purpose

TIRIEXS is provided to allow customization of the input editing behavior for Standard Map screens. This exit is used by any blockmode application containing Standard Map screens and is called for each input screen field.

Arguments

The following table gives a brief description of each of the arguments.

Name	Input/ Output	Description
RUNTIME-PARM1	input	DFHEIBLK (CICS) I/O PCB (IMS) Emulated I/O PCB (TSO)
RUNTIME-PARM2	input	DFHCOMMAREA (CICS) Alternate I/O PCB (IMS) Emulated Alt I/O PCB (TSO)
TIRIEXS-CMCB	input/output	A structure containing the following items:

Name	Input/ Output	Description
TIRIEXS-RETURN-CODE	output	The return value indicating what action the exit wants the runtimes to take for the current screen field. RC-ACCEPT - The exit took no action for the current screen field and accepts the results of Gen's validation. Normal processing will continue. RC-REJECT - The exit requests that the value input in the current screen field be marked in error. This return code should only be used for screen fields that have an edit pattern defined or for screen fields mapped to view attributes that are mandatory or that have permitted values defined. RC-REPROCESS - The exit modified the value that was input in the current screen field (INPUT-VALUE) and requests that this modified input value be revalidated. RC-EXIT-VALUE - The exit modified the value that was stored in the view attribute mapped to the current screen field (ATTRIBUTE-VALUE) and requests that this modified attribute value replace the value that was determined by Gen. This modified attribute value will not be revalidated. RC-ERASE - The exit requests that the value input in the current screen field be erased. This return code should not be used for screen fields mapped to mandatory view attributes.

Name	Input/ Output	Description
TIRIEXS-STATUS-CODE1	input	The status of the validation performed by Gen for the current screen field. SC-OK - The current screen field's value is considered to be valid by Gen's validation routines. SC-FAIL-PENDING - The current screen field's value is considered to be invalid by Gen's validation routines.
TIRIEXS-STATUS-CODE2	input	The processing status of the current screen field. SC-FIRST-PASS - This indicates this is the first time the exit has been called for the current screen field. SC-REENTER - This indicates the exit has been called previously for the current screen field and the exit requested that the field be reprocessed (RC-REPROCESS).
TIRIEXS-ERROR-MSG-NUMBER	input	The error message number determined by Gen's validation routines prior to calling the exit.
INPUT-VALUE	input/output	The value entered in the current screen field. This value should be modified by the exit if the exit returns RC-REPROCESS.
FIELD-LENGTH	input	The length of the current screen field.
FIELD-FILL-CHAR	input	The fill character defined for current screen field.
FIELD-BEGIN-ROWCOL	input	The beginning row and column of the current screen field.
FIELD-END-ROWCOL	input	The ending row and column of the current screen field.
ATTRIBUTE-VALUE	input/output	The value to be stored in the view attribute mapped to the current screen field. This value should be modified by the exit if the exit returns RC-EXIT-VALUE.

Name	Input/ Output	Description
ATTRIBUTE-LENGTH	input	The length of the view attribute mapped to the current screen field.
ATTRIBUTE-TYPE	input	The datatype of the view attribute mapped to the current screen field. ATTR-TEXT - A fixed-length text attribute. ATTR-VARCHAR - A varying-length text attribute. ATTR-NUMERIC - A numeric attribute.
ATTRIBUTE-DECIMAL-PLACES	input	The number of decimal places defined for the view attribute mapped to the current screen field.
ATTRIBUTE-CASE-SENSITIVE	input	The case sensitivity property defined for the view attribute mapped to the current screen field.
MAPNAME	input	The name of the current screen.
MODNAME	input	The modname of the current screen.
DIALECT-NAME	input	The name of the dialect used by the current screen.
DECIMAL-INDICATOR	input	The character used to represent the decimal place in the current dialect.
THOUSANDS-INDICATOR	input	The character used to represent the thousands separator in the current dialect.
CURRENCY-INDICATOR	input	The character used to represent the currency symbol in the current dialect.
TXT-ORIENTATION	input	The orientation of text fields in the current dialect.
NUM-ORIENTATION	input	The orientation of numeric fields in the current dialect.

Name	Input/ Output	Description
EDIT-PATTERN-CLASS	input	The edit pattern class for the current screen field. EPAT-ALPHANUMERIC - An alphanumeric edit pattern. EPAT-NUMERIC - A numeric edit pattern. EPAT-DATE - A date edit pattern. EPAT-TIME - A time edit pattern. EPAT-TIMESTAMP - A timestamp edit pattern. EPAT-NONE - No edit pattern is defined for the current screen field.
FILLER	input	Filler, for future use
GLOBDATA	input	Global data, used internally

Return Code

Update TIRIEXS-RETURN-CODE with the relevant value.

Default Processing

The default processing of this exit is to take no action for the current screen field and return RC-ACCEPT in TIRIEXS-RETURN-CODE.

Customizing the Exit

Copy the TIRIEXS exit to one of your libraries and modify the exit to perform the desired input editing behavior.

Building on z/OS

For information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

TIRIURTX—Ultimate Retry Limit Exit

Ultimate Retry Limit Exit for TSO.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library in member TIRIURTX. The sample exit is written in COBOL and uses standard OS linkage.

The Linkage Parameter list used by TIRIURTX is as follows:

```
01  RUNTIME-PARM1               PIC X.
01  RUNTIME-PARM2               PIC X.
01  ULTIMATE-RETRY-LIMIT        PIC S9(9)   COMP.
01  GLOBDATA                     size 3645 bytes.
```

Purpose

The Ultimate Retry Limit Exit is used by all applications targeting DB2 database on z/OS. The Ultimate Retry Limit Exit allows the user to specify a maximum value for the TRANSACTION RETRY LIMIT system attribute. This value may never be exceeded, either by a SET TRANSACTION RETRY LIMIT statement in an action diagram, or by the Default Retry Limit Exit.

For an explanation of when and how the TRANSACTION RETRY LIMIT system attribute is used see Default Retry Limit Exit in this chapter.

This exit provides a safeguard in case the system attribute TRANSACTION RETRY LIMIT is set to an excessive value by an action diagram. Once the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, no more retries can occur, and the application will fail with a runtime error if the last retry attempt was unsuccessful.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	Input	Emulated IO-PCB
RUNTIME-PARM2	Input	Emulated ALT-IO-PCB
ULTIMATE-RETRY-LIMIT	Input/ output	The absolute limit which is defaulted to 99
GLOBDATA	Input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

If the Ultimate Retry Limit Exit is not modified, the maximum value of TRANSACTION RETRY LIMIT will be 99. The Ultimate Retry Limit Exit may be modified to return a value of zero to suppress all retry attempts.

Customizing the Exit

Copy TIRIURTX exit to one of your libraries or directories. For TSO and IEFAE; applies only when RETRY TRANSACTION statement executes.

Modify the copied exit as needed. When you have completed your modifications, install the exit as described in Customizing and Installing z/OS User Exits.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

z/OS Middleware User Exits—CICS TCP/IP Direct Connect Exits

The CA Gen CICS TCP/IP Direct Connect product consisting of the TILSTNR and TICONMGR has been stabilized and the CICS Sockets Server Listener, TISRVLIS, implementation and the CICS Multi Socket Server Listener are provided.

The following table describes the Assembler exits invoked by the Sockets Server Listener program:

CICS TCP/IP Direct Connect (TISRVLIS): Language: Assembler		
User Exit Name	**Source Code**	**Description**
TIRSLEXT	CEG8SAMP/TIRSLEXT	CICS Sockets Server Listener exit
TIRSLTMX	CEG8SAMP/TIRSLTMX	CICS Sockets Server Listener Timeout exit

Details of the preceding user exits follow. Each one is described in a separate section

TIRSLEXT—CICS Sockets Server Listener Exit

The CA Gen CICS Sockets Server Listener Exit is used by the z/OS CICS user-written Listener programs TISRVLIS and TISRVMSL, the CA Gen TCP/IP implementations.

Note: The TIRSLEXT exit routine is invoked by CICS Sockets Server Listener program TISRVLIS and the CICS Multi Sockets Server Listener program TISRVMSL. The TIRSLEXT exit is not invoked by CA Gen servers.

Source Code

The source code for this exit is in CA Gen CEG8SAMP library, in member TIRSLEXT. The sample exit provided is written in ASSEMBLER and is invoked by TISRVLIS using an EXEC CICS LINK API call.

Data used and returned by TIRSLEXT is passed in the COMMAREA as follows:

1. ASCII transaction code
2. EBCDIC transaction code
3. ASCII userid
4. EBCDIC userid
5. ASCII client code page
6. EBCDIC client code page
7. EBCDIC password
8. Enhanced Security Flag
9. Enhanced Type Flag
10. Check Flag
11. Security Flag
12. Client Port
13. Client IP Address
14. CICS System Id
15. Listener GIVESOCKET/TAKESOCKET TIMEOUT - Seconds
16. Listener GIVESOCKET/TAKESOCKET TIMEOUT – Microseconds
17. Server Termid
18. EXRC

The parameters are listed in the Arguments section.

Purpose

The user ID passed in the commarea section of the data is translated from the client's ASCII to the server's EBCDIC code page. In addition, the exit provides the opportunity to do the following:

- Modify the user ID supplied by the client application before it is translated.
- Validate the supplied, optionally modified, and translated security data through an External Security Manager.
- Verify that an enhanced security buffer exists.
- Perform customer-specific security authorization.
- Use the Port and IP Address data for customer-specific activities.
- Provide a System ID to use to transaction route the server via the SYSID operand of the EXEC CICS START API call.
- Modify the value entered for the GIVTIME parameter in the EZACONFG file. This value is used to populate the TIMEOUT Seconds parameter of the SELECT Sockets API call used by the Listener to wait for the socket to be taken (using the GIVESOCKET/TAKESOCKET process) by the server.
- Provide a value to populate the TIMEOUT Microseconds parameter of the SELECT Sockets API call used by the Listener to wait for the socket to be taken by the server.
- Provide a CICS TERMID to be used to start the server transaction.
- Set the Check flag so that the servers are started with the NOCHECK parameter.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
ASCTRAN	input	The server load module ASCII transaction code.
EBDTRAN	input	The server load module EBCDIC transaction code.
ASCUID	input	The ASCII userid as supplied by the client.
EBDUID	output	The EBCDIC translated version of the ASCII userid sent by the client, modified if required.
ASCCOPG	input	The ASCII code page as supplied by the client.

Name	I/O	Description
EBDCODPG	input	The EBCDIC code page as supplied by the client.
EBDPSW	input	The EBCDIC password as supplied by the client.
ENHSCFLG	input	A flag indicating the presence of enhanced security data.
ENHTPFLG	input	A flag indicating if the security data is encrypted.
CHECKFLG	input/output	A flag indicating whether to use the NOCHECK parameter.
RESERVED	none	Reserved.
SECFLAG	output	A flag indicating the result of the security validation. The following values are supported: ZERO - SFUIDAUT (USERID-IS-AUTHORIZED) 709 – ECTYPINV (ENCRYPTION-FLAG-NOT-SET) 902 - SFINVUID (INVALID-USER-ID) 903 - SFINVPSW (INVALID-PSWD) 904 - SFNTAUTF (NOT-AUTHORIZED-FOR-FUNCTION) 905 - SFSYSUNV (SYSTEM-UNAVAILABLE) 906 - SFUIDSUS (USERID-SUSPENDED) 907 - SFPSWEXP (PSWD-EXPIRED) 908 - SFERROR (INTERNAL-ERROR)
PORT	input	The TCP/IP Port of the client.
IPADDR	input	The TCP/IP Address of the client.
CICS	output	CICS System Id to route the server to.
LGVTIMS	input/output	The Listener GIVESOCKET/TAKESOCKET TIMEOUT value in seconds
LGVTIMM	input/output	The Listener GIVESOCKET/TAKESOCKET TIMEOUT value in microseconds
STERMID	output	CICS TERMID to be used in the server START
EXRC	output	Return code field that is not used

Return Code

No explicit return code value is provided by the TIRSLEXT exit. CICS provides information about the return from the CICS Program Link that indicates if TIRSLEXT exists and its execution was successful. In addition, the values contained in the parameter list addressed by the COMMAREA are returned.

Default Processing

The default processing of this exit is to translate the ASCII user ID to EBCDIC and set the Security Flag to USERID-IS-AUTHORIZED. In addition the exit will return the value received as input for the Listener GIVESOCKET/TAKESOCKET TIMEOUT – Seconds and the CHECKFLG.

Customizing the Exit

This is an exit for TISRVLIS and TISRVMSL, not the server application. The exit is invoked after TISRVLIS or TISRVMSL obtains a copy of the cooperative buffer header sent by each client request and extracts the data required to start the server.

The ASCII user ID, as supplied by the client, can be modified by those users that include special characters (excluding A-Z and 0-9) in their user ID fields. This ASCII user ID is then translated to EBCDIC in the exit. If conversion of the security data to uppercase is required, it can be done in this exit.

The variable 'CICS' can be set to the SYSID of the CICS region where the server is to be routed to. If the value of the variable 'CICS' returned to TISRVLIS is the same as the SYSID of the CICS region where TISRVLIS or TISRVMSL is executing, the EXEC CICS START API command will not specify the SYSID parameter (a local START). If the value of the variable 'CICS' returned is different, the returned value of the variable 'CICS' is used in the START.

The Sockets Server implementation lets the server be routed to a CICS region different from the one where the TISRVLIS or TISRVMSL listener is executing. Routing may be invoked by using the SYSID where the server is to execute, as described above, or by using Distributed Routing. Distributed Routing may be implemented via the DSRTPGM program or the CICS transaction definition.

The CICS Sockets Interface must be active in the CICS region the server is routed to but an active listener is not required in that CICS region.

This exit includes sample code that can be used to validate the user ID and password, the presence of enhanced security data, the presence of encrypted data, or the presence of encrypted and enhanced security data, to modify the Listener Givesocket/Takesocket Timeout value, the value of the STERMID and the value of the CHECKFLG.

On input, the Listener GIVESOCKET/TAKESOCKET Timeout – Seconds value is the value entered for the GIVTIME parameter in the EZACONFG file. This value is used to populate the TIMEOUT Seconds parameter of the SELECT Sockets API call used by the Listener to wait for the socket to be taken (using the GIVESOCKET/TAKESOCKET process) by the server. Since the EZACONFG file does not capture the equivalent Microseconds value, the only opportunity to provide a value for this parameter is in this exit.

If the STERMID variable is updated with a CICS Terminal ID, the Server START TRANSID will include the TERMID parameter. Note that in this case the USERID will not be specified and a server started this way will inherit the USERID of the Listener.

Use the STERMID option if you are migrating from TICONMGR and used this feature in that implementation. The TIRCSGN user exit is provided for the same reason.

On input, the CHECKFLG is set to Y which means to start the server transaction with CHECK. If you wish a remotely started transaction to be started with NOCHECK, set the flag to N. You can also use the EDBTRAN to decide the value of the flag.

Modify the exit to execute the desired validation code.

Building on z/OS

For information about installing the exit, see MKUECTCP in Customizing and Installing z/OS User Exits.

Ensure the new TIRSLEXT module is made available in the DFHRPL concatenation and if applicable issue a NEW COPY command for the TIRSLEXT program in CICS. The Socket Server Listener does not need to be stopped and restarted to use the new version of TIRSLEXT.

Related User Exits

- TIRSLTMX
- TIRSIPEX

TIRSLTMX—CICS Sockets Server Listener TIMEOUT Exit

The CA Gen CICS Sockets Server Listener Timeout Exit is used by the z/OS CICS user-written Listener program TISRVLIS and TISRVMSL, the CA Gen TCP/IP implementation.

Note: The TIRSLTMX exit routine is invoked by CICS Sockets Server Listener program TISRVLIS and TIRSRVMSL. The TIRSLTMX exit is not invoked by CA Gen servers.

Source Code

The source code for this exit is in the CA Gen CEG8SAMP library, in member TIRSLTMX. The sample exit provided is written in ASSEMBLER and is invoked by TISRVLIS using an EXEC CICS LINK API call.

Data used and returned by TIRSLTMX is passed in the COMMAREA as follows:

1. TISRVLIS EBCDIC transaction code
2. CICS APPLID
3. Listener ACCEPT TIMEOUT – Seconds
4. Listener ACCEPT TIMEOUT – Microseconds
5. Listener READ TIMEOUT – Seconds
6. Listener READ TIMEOUT – Microseconds
7. Listener ERROR TIMEOUT – Seconds, error processing
8. Listener ERROR TIMEOUT – Microseconds, error processing
9. Include IP ADDRESS FLAG – flag to add IP address to error message

Purpose

This exit can be used to customize the Listener's SELECT API Calls timeout value that controls the wait time involved with accepting new connections (ACCEPT), reading input data, and reading data for error processing. The corresponding values entered in the EZACONFG file are passed to this exit in the seconds field. These values can remain as they are or can be modified. In addition an extra value may be provided at the microsecond level.

The exit provides the opportunity to do the following tasks:

- Modify the value entered for the ACCTIME parameter in the EZACONFG file. This value is used to populate the TIMEOUT Seconds parameter of the SELECT Sockets API call used by the listener to wait for new connections.
- Provide the value to populate the TIMEOUT Microseconds parameter of the SELECT Sockets API call used by the listener to wait for new connections.
- Modify the value entered for the REATIME parameter in the EZACONFG file. This value is used by the listener to populate the TIMEOUT Seconds parameter of the SELECT Sockets API call used by the TISRVLIS listener when first reading input data. In the TISRVMSL listener, it is used as a wait time.
- Provide the value to populate the TIMEOUT Microseconds parameter of the SELECT Sockets API call used by the listener when first reading input data.
- Modify the value used by the Listener to populate the TIMEOUT Seconds parameter of the SELECT Sockets API call used by the listener when reading input data to process error conditions that require sending an error response to the client.

- Modify the value used by the Listener to populate the TIMEOUT Microseconds parameter of the SELECT Sockets API call used by the listener when reading input data to process error conditions that require sending an error response to the client.
- Modify the value to turn on including the IP Address in some of the socket and CICS error messages.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
SLTRANID	input	The Listener EBCDIC transaction code.
CICSAPPL	input	APPLID of the CICS where Listener runs.
ACCTOUTS	input/output	The Listener ACCEPT TIMEOUT value in seconds
ACCTOUTM	input/output	The Listener ACCEPT TIMEOUT value in microseconds
REATOUTS	input/output	The Listener READ TIMEOUT value in seconds
REATOUTM	input/output	The Listener READ TIMEOUT value in microseconds
ERRTOUTS	input/output	The Listener ERROR TIMEOUT value in seconds, used in error processing
ERRTOUTM	input/output	The Listener ERROR TIMEOUT value in microseconds, used in error processing
IPADRFLG	input/output	The IP address flag

Return Code

No explicit return code value is provided by the TIRSLTMX exit. CICS provides information about the return from the CICS Program Link that indicates if TIRSLTMX exists and its execution was successful. In addition, the values contained in the parameter list addressed by the COMMAREA are returned.

Default Processing

The default processing of this exit is to check if the TISRVLIS trancode matches the EIBTRNID and return.

Customizing the Exit

This is an exit for the TISRVLIS or TISRVMSL listener programs, not the server application. The exit is invoked before TISRVLIS or TISRVMSL accepts a connection from a client and has access to the data sent by the client.

The SLTRANID and the CICSAPPL can be used to decide if the TIMEOUT values provided by the EZACONFG file to be used by the Listener suffice or must be changed.

The Listener Accept TIMEOUT Seconds, as received by this exit contains the value specified in the EZACONFG file for the ACCTIME parameter. This value can be changed, including zeroed out. In addition, if a value smaller than 1 second is required the Listener Accept TIMEOUT Microseconds variable can be used.

The Listener Read TIMEOUT Seconds, as received by this exit contains the value specified in the EZACONFG file for the REATIME parameter. This value can be changed, including zeroed out. In addition, if a value smaller than 1 second is required the Listener Read TIMEOUT Microseconds variable can be used.

When an error condition is encountered that requires the Listener to send an error message to the client, the Listener must first read all the data present at the socket before sending the error message. Part of this read involves a SELECT Sockets API call. The ERRTOUTS and ERRTOUTM values provided in this exit are used to make up this TIMEOUT parameter. The ERRTOUTS value passed to this exit on input is the value specified in the listener code and is zero seconds. The listener code sets the ERROR TIMEOUT - Microseconds to 50. The only opportunity to provide a value for these parameters is in this exit.

The IP Address Flag is set in this exit to turn on the writing of the IP Address as part of some of the socket and CICS error messages. The default is N—do not write the IP Address.

This exit includes sample code that can be used to set the various TIMEOUT parameters as required. Modify the exit to execute the desired code.

Building on z/OS

For information about installing the exit, see MKUECTCP in Customizing and Installing z/OS User Exits.

Ensure the new TIRSLTMX module is made available in the DFHRPL concatenation and if applicable issue a NEW COPY command for the TIRSLTMX in CICS. Stop and restart the Sockets Server Listener to pick up the changes.

Related User Exits

- TIRSLEXT
- TIRSIPEX

z/OS Middleware User Exits—IMS TCP/IP Direct Connect Exits

The CA Gen IMS TCP/IP Direct Connect product provides the following set of user exits. These exits are implemented in Assembler. The following table briefly describes each of the exits:

IMS TCP/IP Direct Connect (CAGRITCP): Language: Assembler		
User Exit Name	**Source Code**	**Description**
TIRxxTD	CEG9SAMP/TIRxxTD	IMS TCP/IP Destination ID exit
TIRxxTDC	CEG9SAMP/TIRxxTDC	IMS TCP/IP Decryption exit
TIRxxTSC	CEG9SAMP/TIRxxTSC	IMS TCP/IP Security exit

Note: The xx in the name denotes the IMS release that was used to assemble the exit. Currently, IMS 10, 11, 12, and 13 are supported.

Details of the preceding user exits follow. Each one is described in a separate section.

TIRxxTD—TCP/IP Destination ID Exit

The CA Gen TCP/IP Destination ID Exit is used by the z/OS IMS Connect User Message Exit CAGRITCP/TIRxxTCP. CAGRITCP is a component of the TCP/IP Direct Connect Option for IMS.

Note: The name TIRxxTCP is used to provide multiple copies of the CAGRITCP exit, each for a different version of IMS Connect. The selected TIRxxTCP exit must be renamed to CAGRITCP before deployment in IMS Connect. The TIRxxTD exit routine is invoked by IBM's z/OS IMS Connect Product. The TIRxxTD exit is not invoked by CA Gen code.

Note: The *xx* in the name denotes the IMS release that was used to assemble the exit.

Source Code

The source code for this exit is in CA Gen CEG9SAMP library, in member TIRxxTD where xx is 10, 11, 12, or 13. The sample exit provided is written in ASSEMBLER and uses standard OS Linkage.

On entry, register usage is as follows:

Register 1—parameter list, mapped to DSECT DESTAREA.

Register 13—address of save area.

Register 14—caller's return address.

Register 15—this exit entry address.

The parameter list used by TIRxxTD is passed in Register 1 as a list of addresses, the last one being indicated by the high-order bit being set on. Each address in the list addresses one argument as follows:

1. Exit Interface Block XIB
2. Client's IP Address
3. Client's Port
4. IMS transaction code
5. User ID as supplied by the Client
6. RACF Groupid
7. Datastore (IMS region) name
8. IMS LTERM
9. Return Code
10. Reason Code

Purpose

This exit is used to provide the name of the IMS region, called Datastore by IMS Connect, of where the request is being sent and the name of an LTERM to be used. The extra data is available for information purposes only.

Arguments

The following table gives a brief description of each argument.

Name	I/O	Description
DESXIB	input	Address of the Exit Interface Block (XIB).
DESIPAD	input	Address of the Client's IP Address.
DESPORT	input	Address of the Client's Port.
DESTRNCD	input	Address of the IMS transaction code.
DESUSRID	input	Address of the userid supplied by the client, translated to EBCDIC.
DESGRPID	input	Address of the RACF Groupid.
DESDSTID	output	Address of the Datastore (IMS region).

Name	I/O	Description
DESLTERM	output	Address of the IMS LTERM.
DESRC	output	Address of the return code. The return code can be: 0 indicates success 2 indicates no active datastore found in XIBDS, 4 indicates other failure
DESRSC	output	Address of the reason code, not used.

Return Code

Register 15 should be set to a value consistent with the value set in the field pointed to by DESRC.

Default Processing

The default processing of this exit is to return the first active datastore name in the XIBDS (XIB Data Store) control block and create an LTERM consisting of the first four bytes of the IMS trancode plus the first four bytes of the client's IP Address. Sets the return code pointed to by DESRC to zero.

This module returns control to the caller using the address passed to it at entry in Register 14. All registers except Register 15 are restored.

Customizing the Exit

This exit can take the data passed to it to determine what IMS region to use for the request being processed and the LTERM to be used.

Ensure DESRC points to the relevant return code. Return control to the caller using the value passed in Register 14 at entry. All registers except Register 15 must be restored. Results are unpredictable if invalid data is returned. Register 15 should contain the return code.

Note: Be sure to use the source that corresponds to your IMS release.

Building on z/OS

For information about installing the exit, see MKUEITCP in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRxxTDC
- TIRxxTSC

TIRxxTDC—TCP/IP Decryption Exit

The CA Gen TCP/IP Decryption Exit is used by the z/OS IMS Connect User Message Exit CAGRITCP/TIRxxTCP. CAGRITCP is a component of the TCP/IP Direct Connect Option for IMS.

Note: The name TIRxxTCP is used to provide multiple copies of the CAGRITCP exit, each for a different version of IMS Connect. The selected TIRxxTCP exit must be renamed to CAGRITCP prior to deployment in IMS Connect. The TIRxxTDC exit routine is invoked by IBM's z/OS IMS Connect Product. The TIRxxTDC exit is not invoked by CA Gen code.

Note: The xx in the name denotes the IMS release that was used to assemble the exit.

Source Code

The source code for this exit is in CA Gen CEG9SAMP library, in member TIRxxTDC where xx is 10, 11, 12, or 13. The sample exit provided is written in ASSEMBLER and uses standard OS Linkage.

On entry register usage is as follows:

Register 1 - parameter list, mapped to DSECT DCRPAREA.

Register 13 - address of save area.

Register 14 - caller's return address.

Register 15 - this exit entry address.

The parameter list used by TIRxxTDC is passed in Register 1 as a list of addresses, the last one being indicated by the high-order bit being set on. Each address in the list addresses one argument as follows:

1. Start of Encrypted buffer
2. Encrypted data length
3. Maximum buffer length
4. Return Code
5. Reason Code

Purpose

This exit is called by TIRxxTCP when it detects that the client sent an encrypted cooperative buffer. This exit can be used to decrypt the area pointed to by DCNCRDAT using the length pointed to by DCNCRDTL. The maximum size of the decrypted data cannot exceed the length pointed to by DCMAXDTL.

The decrypted data is not passed on to the server, instead it is used by TIRxxTCP to extract the cooperative buffer security data to populate the OTMA Security Header and OTMA User Header as determined by SECOTMA flag set by the TIRxxTSC exit.

Arguments

The following table gives a brief description of each argument.

Name	I/O	Description
DCNCRDAT	input	Address of the start of the encrypted buffer.
DCNCRDTL	input	Address of the length of encrypted data.
DCMAXDTL	input	Address of the maximum length available for decrypted data.
DCRC	output	Address of the return code. The return code can be: 0 indicates success 1 indicates exceeded maximum size 2 indicates encryption not used 3 indicates application error 4 indicates other failure
DCRSC	output	Address of the reason code. The reason code can be: 2 indicating not used

Return Code

Register 15 should be set to a value consistent with the value set in the field pointed to by DCRSC.

Default Processing

This exit is invoked only when the received cooperative request is encrypted. The default action of this exit is to issue a return code of not used in DCRC and a reason code of not used in DCRSC, to indicate a problem.

This module returns control to the caller using the address passed to it at entry in Register 14. All registers except Register 15 are restored.

Register 15 contains the return code of 2.

Customizing the Exit

Modify the exit to invoke the required decrypt algorithm, set the return, and reason codes accordingly. The decrypted data returned cannot exceed the value pointed to by DCMAXDTL. Ensure DCRC points to the relevant return code. Return control to the caller using the value passed in Register 14 at entry. All registers except Register 15 must be restored. Results are unpredictable if invalid data is returned.

Note: Be sure to use the source that corresponds to your IMS release.

Building on z/OS

For information about installing the exit, see MKUEITCP in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRxxTD
- TIRxxTSC

TIRxxTSC—TCP/IP Security Exit

The CA TCP/IP CA Gen Security Exit is used by the z/OS IMS Connect User Message Exit CAGRITCP/TIRxxTCP. CAGRITCP is a component of the TCP/IP Direct Connect Option for IMS.

Note: The name TIRxxTCP is used to provide multiple copies of the CAGRITCP exit, each for a different version of IMS Connect. The selected TIRxxTCP exit must be renamed to CAGRITCP before deployment in IMS Connect. The TIRxxTSC exit routine is invoked by IBM's z/OS IMS Connect Product. The TIRxxTSC exit is not invoked by CA Gen code.

Note: The xx in the name denotes the IMS release that was used to assemble the exit.

Source Code

The source code for this exit is in CA Gen CEG9SAMP library, in member TIRxxTSC where xx is 10, 11, 12, or 13. The sample exit provided is written in ASSEMBLER and uses standard OS Linkage.

On entry register usage is as follows:

Register 1 - parameter list, mapped to DSECT SECAREA.

Register 13 - address of save area.

Register 14 - caller's return address.

Register 15 - this exit entry address.

The parameter list used by TIRxxTSC is passed in Register 1 as a list of addresses, the last one being indicated by the high-order bit being set on. Each address in the list addresses one argument as follows:

1. OTMA Security Flag
2. Client's IP Address
3. Client's Port
4. IMS transaction code
5. Client Code Page
6. Data Type
7. Exit Interface Block XIB
8. Length of user data
9. CFB Userid
10. CFB Password
11. RACF Groupid
12. Return Code
13. Reason Code

Purpose

This exit is used to enable the selection of security type for the OTMA Headers being formatted by the TIRxxTCP IMS Connect User Message exit. The TIRxxTSC exit is passed to various items that can be used to determine the appropriate security type or the exit can do security checking itself and return the appropriate values to match the required OTMA security.

Arguments

The following table gives a brief description of each argument.

Name	I/O	Description
SECOFLG	output	Address of OTMA security flag.
SECIPAD	input	Address of the Client's IP Address.
SECPORT	input	Address of the Client's Port.
SECTRNCD	input	Address of the IMS transaction code.
SECCODPG	input	Address of the Client Code Page.
SECDTKEY	input	Address of data type. The data type can be: 0 for ASCII or 1 for EBCDIC. It is 1 since the security data has been translated.
SECXIB	input	Address of the Exit Interface Block (XIB)
SECDATL	input	Address of the length of the security data.
SECUID	input	Address of the userid as supplied by the Client.
SECPSW	input	Address of the password as supplied by the Client.
SECGRPID	output	Address of the RACF Groupid.
SECRC	output	Address of the return code. The return code can be: 0 indicates success 2 indicates security not used 3 indicates application error 4 indicates other failure
SECRSC	output	Address of the reason code, not used.

Return Code

Register 15 should be set to a value consistent with the value set in the field pointed to by SECRC.

Default Processing

The default processing of this exit sets the OTMA security flag pointed to by SECOFLG to OSECNON for none and sets the return code pointed to by SECRC to zero.

This module returns control to the caller using the address passed to it at entry in Register 14. All registers except Register 15 are restored.

Customizing the Exit

This exit can use the data passed to it to determine what the SECOTMA variable should be set to. This variable needs to match the OTMASE parameter configured for OTMA in the IMS region.

Set SECOTMA to F or C to cause TIRxxTCP to populate the OTMA Security Header with provided user ID, group ID, or token and the OTMA User Header Passticket with the provided password.

Set SECOTMA to N to cause TIRxxTCP to not populate the OTMA Security and User Headers with security data.

The security data used is the user ID and password provided in the security offset of the cooperative buffer if enhanced security is used, the user ID and password provided in the cooperative buffer header if standard security is used and blanks if none is provided. However if SECOTMA is set to a value other than N and no security data is provided the IMS Connect User is used as the user ID, no password will be provided and this can result in a failure from OTMA with a reason code of SECFNPUI, which will be interpreted as an XERR with a code of 0902, message 'Security: Invalid User ID'.

With the changes to support IPv6 the length of the TCP/IP IP Address has changed. The SECIPAD parameter contains the address of an IP Address that is 4 bytes in length for IPv4 implementations and is 16 bytes in length for IPv6 implementations. This is further represented in the SECIP DSECT.

Modify the exit as required. Ensure SECRC points to the relevant return code. Return control to the caller using the value passed in Register 14 at entry. All registers except Register 15 must be restored. Results are unpredictable if invalid data is returned. Register 15 should contain the return code.

Note: Be sure to use the source that corresponds to your IMS release.

Building on z/OS

For information about installing the exit, see MKUEITCP in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRxxTD
- TIRxxTDC

z/OS Middleware User Exits – WebSphere MQ CICS

WebSphere MQ Transaction Dispatcher for CICS (TDC) Exit

The CA Gen WebSphere MQ Transaction Dispatcher product provides the following user exit. This exit is implemented in COBOL. The following table briefly describes this exit:

WebSphere MQ TDC: Language: COBOL		
User Exit Name	**Source Code**	**Description**
TIRMQTDZ	CEHESAMP/TIRMQTDX	TDC Parameter exit

Details of the preceding user exit follow, described in a separate section.

TIRMQTDX—WebSphere MQ Transaction Dispatcher for CICS Parameter Exit

The Parameter Exit for the CA Gen WebSphere MQ Transaction Dispatcher for CICS (TDC) option is used by z/OS WebSphere MQ servers.

For each trigger event that initiates the TDC, the TDC sends a copy of the parameter list to the TDX (TIRMQTDX) in the DFHCOMMAREA. The TDX sends the list back to the TDC including any parameter modifications programmed into the TDX. The TDC accepts the parameter list without checking for invalid entries.

Note: The TDX is responsible for validating any parameter changes. If you change the default behavior of the exit, ensure that it passes a correct parameter, because the TDC does not validate parameters it receives from the TDX.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRMQTDX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list provided is as follows:

```
01  DFHCOMMAREA.
    03  TDC-CICS-SYSID                 PIC X(4).
    03  TDC-MQS-QUEUENAME              PIC X(48).
    03  TDC-PARMLIST.
        05  TDC-MAX-MPL                PIC S9(8) COMP.
        05  TDC-Q-CHECK-FREQ           PIC S9(8) COMP.
        05  TDC-Q-DEPTH-PER            PIC S9(8) COMP.
        05  TDC-SLOW-AT-MXT            PIC S9(8) COMP.
        05  TDC-SLOW-INTERVAL          PIC S9(8) COMP.
        05  TDC-MQGET-WAITTIME         PIC S9(8) COMP.
        05  TDC-OVERIDE-USERID         PIC X(8).
        05  TDC-VSAM-DDNAME            PIC X(8).
        05  TDC-CICS-MSGDEST           PIC X(4).
        05  TDC-REQID-PREFIX           PIC X(2).
        05  TDC-TSQ-KEY-PREFIX         PIC X(2).
        05  TDC-SECURITY-MODE          PIC X(1).
        05  TDC-MESSAGE-LEVEL          PIC X(1).
        05  TDC-TEMPSTOR-METHOD        PIC X(1).
        05  TDC-REPTOQM-SELECT         PIC X(1).
        05  TDC-TASK-TABLE-MAX         PIC S9(8).
        05  TDC-CHILD-MQGET-WAIT               PIC S9(8).
        05  TDC-RESERVED               PIC X(4).
    03  TDC-PROCESS-OBJ-UA             PIC X(128).
```

Purpose

This exit contains parameters that are used by the MQSeries TDC program TIRMQTDC. These parameters can be modified to influence how the parent TDC process behaves.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
DFHCOMMAREA	input/output	CICS Communication area with the following contents.
TDC-CICS-SYSID	input	CICS SYSID for this invocation. Do not modify.
TDC-MQS-QUEUENAME	input	Name of MQ Queue being processed. Do not modify.
TDC-PARMLIST	input/output	Parameter list structure with the following items:
TDC-MAX-MPL	input/output	Limits the total number of child TDC processes that the TDC spawns against the MQS queue during heavy loads. Default: 5 The valid range is 1 through the CICS MXT value. Remember that WebSphere MQ dispatches on a maximum of 8 TCBs.

Name	I/O	Description
TDC-Q-CHECK-FREQ	input/output	Sets the number of messages the TDC processes before checking queue depth. Accepts a numeric value. Default: 10 The valid range is 1 through 4095
TDC-Q-DEPTH-PER	input/output	Sets the queue depth threshold at which the TDC spawns a child TDC process. Default: 20 The valid range is 1 through 4095.
TDC-SLOW-AT-MXT	input/output	Enter slowdown mode when active tasks reach CICS MXT minus TDC-SLOW-AT-MXT. The TDC enters slowdown mode if it detects a Short-On-Storage condition, or if the current active task count exceeds CICS MXT minus the TDC-SLOW-AT-MXT value. The TDC exits slowdown mode whenever these conditions are resolved. **Note:** Setting TDC-SLOW-AT-MXT high may severely affect response time. Default: 20 The valid range is 1 through CICS MXT value.
TDC-SLOW-INTERVAL	input/output	Defines the amount of time that the TDC waits between starting server managers while in slowdown mode, in seconds. **Default:** 2 The valid range is 1 through 359999.
TDC-MQGET-WAITTIME	input/output	Determines the persistence of the parent TDC, the length of time the TDC holds the MQS queue open waiting for a message to arrive, in milliseconds. Setting this to a high value reduces triggering and transaction overhead, but the task remains suspended on the task chain for the duration if no messages are available. This value affects only the parent invocation of the TDC. Spawned child TDCs wait for messages for one second regardless of this setting. **Default:** 5000 The valid range is 0 through 268435455 milliseconds
TDC-OVERIDE-USERID	input/output	Override user ID for CA Gen transactions. The default is blank, but you can set it to any valid user ID. If you use this parameter, set the TDC-SECURITY-MODE parameter to O.
TDC-VSAM-DDNAME	input/output	VSAM file DDNAME. By default, this parameter is TITDTEMP, but you can set it to any CICS-supported 8 character name. Used when the TDC-TEMPSTOR-METHOD parameter is set to V.

Name	I/O	Description
TDC-CICS-MSGDEST	input/output	Controls where the TDC directs status messages. **Default:** CSSL The valid range is any CICS-supported eight characters DDNAME.
TDC-REQID-PREFIX	input/output	2-byte REQID prefix (Start REQID) that supports any CICS-supported two characters. By default, the value is SQ.
TDC-TSQ-KEY-PREFIX	input/output	2-byte temporary storage prefix that supports any CICS-supported two characters. By default, the value is RQ. Used when the TDC-TEMPSTOR-METHOD parameter is set to M or A.
TDC-SECURITY-MODE	input/output	Security mode for CA Gen transactions started by the TDC. Accepts the following values: **D**—(default) CA Gen transactions inherit the user ID of the dispatcher transaction. **C**—CA Gen transactions are started with the user ID supplied in the message. The user ID of the dispatcher must be an authorized surrogate for the message user ID. Clients must be using at least Advantage Gen 6.5 or later. **O**—CA Gen transactions are started with the user ID supplied by the value in the TDC-OVERIDE-USERID parameter. The user ID of the dispatcher must be an authorized surrogate for the user ID supplied in the TDC-OVERIDE-USERID parameter. **Note:** For more information about these modes, see the *Distributed Processing – WebSphere MQ User Guide*.
TDC-TEMPSTOR-METHOD	input/output	Sets the temporary storage method. Accepts the following values: **M**—(default) Use CICS main temporary storage (TSQ). **A**—Use auxiliary temporary storage method (TSQ). **V**—Use VSAM file control (SDT recommended). If you set this parameter to V, you also need to set TDC-VSAM-DDNAME.
TDC-REPTOQM-SELECT	input/output	Use Client Specified (C) or Local (L) Manager for Reply to Queue. Client Specified is the default.
TDC-TASK-TABLE-MAX	input/output	Maximum table occurence for started servers.
TDC-CHILD-MQGET-WAIT	input/output	Time to keep the child queue open.
TDC-RESERVED	input	Reserved for future use.
TDC-PROCESS-OBJ-UA	input	Copy of the 128-byte process definition user area associated with a queue and passed into the trigger record. Not modifiable. The TDC cannot use this area, but the exit may read a value placed in the user area for input during an installation.

Return Code

No explicit return code value is provided by the TIRMQTDX. CICS provides information about the return from the CICS Program Link that indicates if TIRMQTDX exists and its execution was successful. In addition, the values contained in the parameter list are returned.

Default Processing

The default processing of this exit is to return without modifying any of TDC parameters.

Customizing the Exit

This program is invoked during the parent TIRMQTDC invocation process. The parameter list, TDC-PARMLIST, can be modified to influence the behavior of program TIRMQTDC. Additionally the CICS Sysid, the name of the MQS Queue being processed, and a copy of the process Object User Area are supplied. The CICS Sysid and the MQ Queue name must not be modified.

The parameter list provided in the Commarea contains the default values on entry. These values can be altered as required but care must be taken to insure that a valid parameter list is returned to TIRMQTDC. Results are unpredictable if invalid values are inserted in the Commarea.

Note: For more information about the TDC, see the *Distributed Processing—WebSphere MQ User Guide.*

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

z/OS Server User Exits—CICS

TIRTIARX—DB2 Message Exit

z/OS Server Managers use the CA Gen DB2 Message Exit.

Source Code

The source code for the version of the exit used by CICS application is in CA Gen CEHBSAMP library, in member TIRCTIAX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRTIARX is as follows:

```
01  RUNTIME-PARM1              PIC X.
01  RUNTIME-PARM2              PIC X.
01  TIRFAIL-SQLCA              PIC X.
01  TIRTIAR-ERRORS             PIC X.
01  TIRTIAR-TEXT-LEN           PIC X.
01  GLOBDATA                   size 3645 bytes.
```

Purpose

The DB2 Message Exit is used by all applications targeting DB2 database on z/OS. The TIRFAIL subroutine of the Dialog Server calls the DB2 Message exit, TIRTIARX, whenever an unrecoverable DB2 failure occurs. TIRTIARX then calls the subroutine DSNTIAR to convert the SQL code into text. The messages returned by DSNTIAR are then merged with the runtime error messages.

TIRTIARX exit must be a DLL in order to be invoked by Gen applications, even by those using Compatibility option. DSNTIAR and DSNTIAC are provided by IBM as non-DLL modules. Therefore they need to be invoked by via TIRLGLOD.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included by the Translator.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included by the Translator.
TIRFAIL-SQLCA	input	Pointer to SQLCA
TIRTIAR-ERRORS	input	Pointer to TIRTIAR-ERRORS structure
TIRTIAR-TEXT-LEN	input	Length of one text line
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Behavior

As provided by CA Gen, the default exit dynamically calls DSNTIAR and is compatible with prior releases. However the sample code also contains examples of how to call DSNTIAR or DSNTIAC statically.

The call to TIRTIARX is made when TIRFAIL is building the table of messages and occurs prior to calling the default termination exit. For more information, see the Online Termination Exit and Batch Termination Exit.

Customizing the DB2 Message Exit

Copy the default exit to one of your own libraries. The member name is TIRCTIAX. The default exit includes example code for the four possible combinations of calls. There are dynamic and static calls of both DSNTIAR and of DSNTIAC. Simply comment out the default call and remove the comments from the one you want to use.

To statically call DSNTIAR or DSNTIAC, link the routine into the TIRCTIAZ DLL not the Gen application.

To dynamically call DSNTIAR or DSNTIAC build this routine as a non-DLL stand-alone executable and provide a CICS program definition (PPT) for it. This means that you need PPT definitions if you use the default TIRTIARX module. If TIRTIARX is customized to call DSNTIAC instead of DSNTIAR see IBM's CICS and/or DB2 documentation about using DSNTIAC.

When you have completed your modifications, install your exit.

Note: DSNTIAC is shipped as source code and must be assembled. If you intend to use it, see your DB2 or CICS systems programmer to ensure that is has been assembled and that a load module is available. If not, either the install of the application module will fail with an unresolved module at the time of the link if your call is static, or you will abend at runtime if your call is dynamic.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRCDPTX—Dynamic Plan TSQ Processing Exit

z/OS Server Managers use the CA Gen Dynamic Plan TSQ Processing Exit.

Source Code

This exit is used by CICS applications only. The source code for this exit is in CA Gen CEHBSAMP library, in member TIRCDPTX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRCDPTX is as follows:

```
01  RUNTIME-PARM1          PIC X.
01  RUNTIME-PARM2          PIC X.
01  Q-NAME                 PIC X(8).
01  ACTION-CODE            PIC 9.
01  GLOBDATA                    size 3645 bytes.
```

Purpose

This exit is called when the delete of the TSQ used by the Dynamic Plan Exit (TIRC$EXT) fails because the TSQ does not exist. It is used to return a flag to control how the runtime handles the missing TSQ condition.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if Translated.
Q-NAME	input	The name of the TSQ which we expected to find, but is missing.

Name	I/O	Description
ACTION-CODE	input/output	A 1 byte numeric field indicating how the runtime should handle the missing TSQ condition: 1 – Abend and rollback. 2 – Send an error message to CICS CSSL output and terminate normally, without a rollback. 3 – Send an error message to CICS CSSL output and terminate without an abend but with a rollback. 4 – Ignore the condition and terminate without an abend or rollback.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

The default processing of this exit is to return an ACTION-CODE of 2 which causes the runtime to handle the condition encountered by the CICS API command that deletes the TSQ, send a message to the CICS CSSL output and continue processing without rolling back any database changes done by the application.

Customizing the Exit

Copy the TIRCDPTX exit to one of your libraries and modify ACTION-CODE to return a value other than 2.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None

TIRSRTRX—Default Retry Limit Exit Processing

z/OS Server Managers for CICS cooperative applications use the CA Gen Default Retry Limit Exit.

TIRSURTX—Server Ultimate Retry Limit Exit

TBD

Purpose

The Ultimate Retry Limit Exit allows the user to specify a maximum value for the TRANSACTION RETRY LIMIT system attribute. This value may never be exceeded, either by a SET TRANSACTION RETRY LIMIT statement in an action diagram, or by the Default Retry Limit Exit.

For an explanation of when and how the TRANSACTION RETRY LIMIT system attribute is used see Default Retry Limit Exit.

Once the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, no more retries can occur, and the application will fail with a runtime error if the last retry attempt was unsuccessful.

TIRSYSIX—System ID Exit

z/OS Dialog Managers use the CA Gen System Identification Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library in member TIRCSYSX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRSYSIX is as follows:

```
01  LOCAL-SYSTEM-ID             PIC X(8).

01  GLOBDATA                    size 3645 bytes
```

This exit contains CICS API calls, which require it to be processed by the Translator. The Translator automatically includes data structures for DFHEIBLK and DFHCOMMAREA in the place of RUNTIME-PARM1 and RUNTIME-PARM2 thus RUNTIME-PARMx are not specified.

Purpose

The purpose of TIRSYSIX is to enable logic that lets the same application be implemented on multiple systems and perform processing specific to each system targeted.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
DFHEIBLK	input	Automatically included by the Translator.
DFHCOMMAREA	input	Automatically included by the Translator.
LOCAL-SYSTEM-ID	output	The identifier of the system where the application is executing.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

The default processing of the exit is to issue the CICS Assign Sysid command to retrieve the system ID. If successful, the retrieved ID is returned; otherwise, the literal CICS is returned.

Customizing the Exit

Copy the TIRCSYSX member to one of your libraries and modify to populate the LOCAL-SYSTEM-ID as required by the application.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRALLOX
- TIRPROUX

TIRUSRIX—User ID Exit

z/OS Dialog Managers use the CA Gen User Identification Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library in member TIRCUSRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRUSRIX is as follows:

```
01  FILLER-PARM                      PIC X.
01  TIRUSRID-PARM.
    05  IET-USER-ID                   PIC X(8).
    05  IET-USER-ID2                  PIC X(8).
01 GLOBDATA                          size 3645 bytes.
```

This exit contains CICS API calls, which require it to be processed by the Translator. The Translator automatically includes data structures for DFHEIBLK and DFHCOMMAREA in the place of RUNTIME-PARM1 and RUNTIME-PARM2 thus RUNTIME-PARMx are not specified.

Purpose

The purpose of TIRUSRIX is to obtain the userid and terminal ID of the executing application so that these values can be used as part of the key for the DB2 Profile Table and in the application itself.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description		
DFHEIBLK	input	Automatically included by the Translator.		
DFHCOMMAREA	input	Automatically included by the Translator.		
FILLER-PARM	input	Not used.		
TIRUSRID-PARM	output	A pointer to a structure containing the following items:		
		IET-USER-ID	output	The userid to be used in the application.
		IET-USER-ID2	output	The ID to be used as part of the Profile Table key.

Return Code

No explicit return code value is set by the user exit.

Default Processing

There are two possible implementations for this exit.

The default processing of the exit is coded for applications that execute with a terminal facility, these are blockmode and servers that use SNA and ECI. In this case, the exit checks that the terminal ID and user ID values are present and these values are returned. If only the terminal ID is present, its value is returned as both terminal ID and user ID. If there is no terminal ID value, the CICS Task ID is returned as both terminal ID and user ID.

The exit also contains sample code that can be used for applications that execute without a terminal facility, these are servers that use TCP/IP and MQSeries. For these applications, if the user ID and terminal ID are present the exit returns these values. If only the user ID is present, it is returned and the CICS Task ID is returned for the terminal ID. If only the terminal ID is present, its value is returned for both fields. If neither user ID nor terminal ID is present, the CICS Task ID is returned for both.

Customizing the Exit

Copy the TIRUSRIX to one of your libraries and modify to populate either IET-USER-ID or IET-USER-ID2 as required by the application.

Note: IET-USER-ID is used by the application as its User Identifier while IET-USER-ID2 is used as part of the Key to the RPROF (Profile Manager) Table.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

TIRSECRX

TIRSECRX—Security Interface Exit

z/OS Dialog Managers use the CA Gen Security Interface Exit.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRSECRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                          PIC X.
01  RUNTIME-PARM2                          PIC X.
01  TIRSECR-CMCB.
    03  TIRSECR-USERID                     PIC X(8).
    03  TIRSECR-TRANCODE                   PIC X(8).
    03  TIRSECR-TERMINAL-ID                PIC X(8).
    03  TIRSECR-SYSTEM-ID                  PIC X(8).
    03  TIRSECR-LOAD-MODULE                PIC X(8).
    03  TIRSECR-PSTEP-NAME                 PIC X(32).
    03  TIRSECR-DIALECT                    PIC X(32).
    03  TIRSECR-RETURN-CODE                PIC XX.
    03  TIRSECR-FAILURE-MSG                PIC X(80).
01 GLOBDATA                                size 3645 bytes.
```

Purpose

This exit is called by both cooperative and non-cooperative applications to allow transaction-level security checking to be implemented.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description	
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.	
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.	
TIRSECR-CMCB	input/ output	A structure containing the following items:	
	TIRSECR-USERID	input	The userid under which this transaction is executing, as provided by the TIRUSRIX exit.

Name	I/O	Description	
	TIRSECR-TRANCODE	input	The load module transaction code.
	TIRSECR-TERMINAL-ID	input	The terminal ID used by this transaction, spaces if this is a non-terminal transaction.
	TIRSECR-SYSTEM-ID	input	The system ID where this transaction is executing, as provided by the TIRSYSIX exit.
	TIRSECR-LOAD-MODULE	input	The load module name.
	TIRSECR-PSTEP-NAME	input	The Procedure Step name.
	TIRSECR-DIALECT	input	The dialect used by this application.
	TIRSECR-RETURN-CODE	output	A 2-byte character field returning the result of the security check. The following values are supported: SPACES—TIRSECR-ALL-OK Anything else—failure
	TIRSECR-FAILURE-MSG	output	An 80-byte character field, to be populated by this exit, to describe the failure with a message of choice.

Return Code

Update TIRSECR-RETURN-CODE with the relevant value.

Default Processing

The default processing of this exit is to do no security checking and to return TIRSECR-ALL-OK as the return code.

Customizing the Exit

This exit is called by both cooperative and non-cooperative applications.

The cooperative Server Manager calls the Security Interface Exit when a transaction is started. The Server Manager also calls TIRSECVX to provide application-level security. On a Server-to-Server flow, both exits are also invoked. The TIRSECRX exit can be used to check if the current application has authority to invoke the target server trancode.

For cooperative applications, the TIRSECR-USERID is the userid the server transaction is executing under which may or may not be different from the CLIENT_USER_ID sent by the client application.

Modify the exit to perform security checking as required by the application. Ensure that TIRSECR-RETURN-CODE is set to spaces when the security check is successful or some other value to indicate failure. If a message describing the violation is returned in TIRSECR-FAILURE-MSG, the Server Manager will pass it to TIRELOGX.

Building on z/OS

For information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRUSRIX
- TIRSECVX
- TIRELOGX
- TIRTERMA

TIRQCNTX—TSQ Profile Manager Exit

z/OS Dialog Managers use the CA Gen TSQ Profile Manager Exit.

Source Code

The source code for this exit is in the Gen CEHBSAMP library in member TIRQCNTX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The parameter list used by TIRQCNTX is as follows

```
01 RUNTIME-PARM1                    PIC X.
01 RUNTIME-PARM2                    PIC X.
01  TIRQCNTL-CMCB.
    05  TIRQCNTL-QUEUE-NAME         PIC X(8).
    05  TIRQCNTL-STORAGE-TYPE       PIC X.
01 GLOBDATA                         size 3645 bytes.
```

Purpose

This exit is used by CICS applications only. The purpose of the TIRQCNTX is to allow the user to override the name of the queue used for the temporary storage queue profile and the type of storage used for the queue.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.
TIRQCNTL-QUEUE-NAME	input/ output	Name of the temporary storage queue used for the profile manager.
TIRQCNTL-STORAGE-TYPE	input/ output	Type of storage where the queue will reside.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

The default action for the exit is to leave the TSQ Profile Manager queue name , that consists of an internal application ID and the LTERM ID, unchanged and set the default storage type to MAIN.

Customizing the Exit

Copy the TIRQCNTX exit to one of your libraries and modify as required. The exit does not use CICS commands so it does not need to be translated for CICS and it specifies the RUNTIME-PARM1 and RUNTIME-PARM2 in both the Linkage Section and the Procedure Division statement. Ensure these are removed if CICS API calls are added.

The TIRQCNTL-QUEUE-NAME needs to be unique per CICS region.

The TIRQCNTL-STORAGE-TYPE can be set to either USE-AUXILIARY-STORAGE instead of USE-MAIN-STORAGE.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRUPPRX—Uppercase Translation Exit

z/OS Dialog Managers use the CA Gen Uppercase Translation Exit. This exit is also called the Lower-to-Uppercase Conversion Exit.

Source Code

This exit is used by single byte and double byte applications. When used by double byte applications an alternate entry point TIRUPDBx is used. The source code for this for this exit is in CA Gen CEHBSAMP library in member TIRUPPRX. The sample exit is written in COBOL and uses OS linkage.

The Parameter list used by TIRUPPRX is as follows:

```
01  RUNTIME-PARM1                          PIC X.
01  RUNTIME-PARM2                          PIC X.
01  XLATE-TABLE-NAME                      PIC X(8).
01  XLATE-LEN                             PIC S9(4) COMP.
01  XLATE-DATA                            PIC X(4096).
01  GLOBDATA                               size 3645 bytes.
```

Purpose

The purpose of the Uppercase Translation User Exit is to translate character input from lowercase to uppercase. It contains a table of paired lower and uppercase characters. This exit is called by the Dialog Manager to translate the lower caser trancode to upper case, by the TIRFUPPR Function to translate the designated data to upper case and by the Standard Map runtime to translate the identified input data to upper case.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included if translated.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included if translated.
XLATE-TABLE-NAME	input	Name of the translation table to be used.
XLATE-LEN	input	Length of data to be translated.
XLATE-DATA	input/output	Data to be translated.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code value is defined for this exit.

Default Processing

The default processing of this exit is to convert lower case characters to upper case using a table named DEFAULT that contains the English character set(A-Z).

Customizing the Exit

Copy the default exit from the CA Gen CEHBSAMP library to one of your own libraries. The member name is TIRUPPRX.

The exit supports both single byte and double byte languages. Adding support for DBCS is done in the same way as for single byte.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRXINFO—National Language Information Exit

z/OS servers use the CA Gen National Language Information Exit.

Source Code

The source code for this exit can be found in CA Gen CEHBSAMP library, in member TIRXINFO. The sample exit provided is written in Assembler and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
OSID DS A
CODEPGID DS A
PADCHAR DS A
```

Purpose

This exit defines the Code Page Id used to select the National Language translate tables from the list of provided GXTables for all cooperative applications. It also provides the Pad Character applicable to the selected Code Page Id and the Operating System.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
OSID	output	Pointer to the Operating System.
CODEPGID	output	Pointer to the Code Page identifier.
PADCHAR	output	Pointer to the Pad Character.

Return Code

None, the modified values are returned.

Default Processing

The Operating System is set to MVS, the Code Page is set to 037 (US EBCDIC) and the Pad Character is set to hex '40' (blank) which is the relevant pad character for code page 037.

Customizing the Exit

The values contained in variables DEFOPSYS, DEFCODEP and DEFPADCH are loaded into the XINFPARM DSECT which is the Parmlist returned by this exit. Modify the DEFCODEP and DEFPADCH values to indicate the required Code Page and relevant Pad Character. Ensure that for z/OS the DEFOPSYS is set to MVS.

Related User Exits

None

TIRSECVX—Server Client Security Validation Exit

z/OS servers use the CA Gen Server Client Security Validation Exit.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRSECV. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01 RUNTIME-PARM1                                    PIC X.
01 RUNTIME-PARM2                                    PIC X.
01 ENHANCED-SECURITY-FLAG                           PIC X.
01 TIRSECV-CMCB.
03 CLIENT-USERID                                    PIC X(64).
03 CLIENT-PASSWORD                                  PIC X(64).
03 SECURITY-TOKEN-LEN                       PIC 9(09) COMP.
03 SECURITY-TOKEN-PTR POINTER.
03 TIRSECV-TRANCODE                                 PIC X(08).
03 TIRSECV-RETURN-CODE                      PIC X(02).
03 TIRSECV-FAILURE-MSG                      PIC X(80).
```

Purpose

This exit is called for every cooperative flow. To facilitate security validation, a flag indicating whether the security data is for a standard or enhanced buffer has been added to the exit's Parameter List.

The Server Manager calls TIRSECV to allow the Enhanced Security data supplied by the client application to be validated. This exit is intended to provide the opportunity to validate Enhanced Security data. The default code in the exit can be used by applications designed to use Standard Security.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	Under CICS, this is CICS EXEC Interface Block - DFHEIBLK.
RUNTIME-PARM2	input	Under CICS, this is DFHCOMMAREA.
ENHANCED-SECURITYFLAG	input	A 1-byte field indicating whether Enhanced Security is intended for this request.
TIRSECV-CMCB	input/ output	A structure containing the following items:
CLIENT-USERID	input	A 64-byte field containing a copy of the Userid sent by the client.
CLIENT-PASSWORD	input	A 64-byte field containing a copy of the Password sent by the client.
SECURITY-TOKEN-LEN	input	The length of the security token sent by the client.
SECURITY-TOKEN-PTR	input	A pointer to a security token sent by the client.

Name	I/O	Description
TIRSECV-TRANCODE	input	The server load module transaction code.
TIRSECV-RETURNCODE	output	A 2-byte character field returning the result of the validation attempt. The following values are supported: SECURITY-USED—defined as SPACES SECURITY-NOT-USED—defined as "02" SECURITY-APPLICATION-ERROR— defined as "03"
TIRSECV-FAILURE-MSG	output	An 80-byte character field, to be populated by this exit, to describe the failure with a message of choice. This failure message will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of SECURITY-APPLICATIONERROR.

Return Code

See TIRSECV-RETURN-CODE in Arguments.

Default Processing

To provide the opportunity to validate security data while at the same time not impacting those using standard security. The default processing provided for this exit handles 2 possible conditions:

For buffers containing Standard Security data the Client-Userid, Client- Password and Security Token fields are expected to be blank. The default processing is for the exit to return Security-Used, thus indicating that the request is authorized.

For buffers containing Enhanced Security data the Client-Userid, Client-Password and Security Token fields can or cannot contain data. The default processing is for the exit to return Security-Not-Used, thus indicating that no validation processing was attempted.

Customizing the Exit

The exit returns Security-Used when it assumes Standard Security is used. If the intent is to use Enhanced Security, modify the exit to return SECURITYAPPLICATION-ERROR.

The exit returns Security-Not-Used when it assumes Enhanced Security is used. Modify the exit to validate the security data and set the relevant return code. Return SECURITY-USED for an authorized user and SECURITYAPPLICATION-ERROR for a non-authorized user.

When a return code of SECURITY-APPLICATION-ERROR is used an optional failure message can be supplied that will be presented to the client. Optionally, the Security Token can be used for authentication.

Building on z/OS

For information about installing the exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

WRSECTOKEN

TIRDCRYX—Server Decryption Exit

The CA Gen Server Decryption Exit is used by z/OS servers.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRDCRYX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                          PIC X.
01  RUNTIME-PARM2                          PIC X.
01  TIRDCRYP-CMCB.
    03  DATA-BUFFER-PTR                    POINTER.
    03  BUFFER-SIZE                        PIC 9(09) COMP.
    03  DECRYPTION-MAX-SIZE                PIC 9(09) COMP.
    03  TIRDCRYP-RETURN-CODE               PIC X(02).
    03  FAILURE-MSG                        PIC X(80).
```

Purpose

TIRDCRYX is called by the Server Manager after it detects that the client has sent an encrypted cooperative buffer. The user is responsible for decrypting the area pointed to by DATA-BUFFER-PTR for the length BUFFER-SIZE bytes. Maximum size of the decrypted data cannot exceed DECRYPTION-MAX-SIZE.

Note: Encryption/decryption of the request and response cooperative buffers are mutually exclusive - encryption/decryption of the request buffer does not necessitate that the same be done to the response buffer and vice-versa.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	Under CICS, this is CICS EXEC Interface Block - DFHEIBLK.
RUNTIME-PARM2	input	Under CICS, this is DFHCOMMAREA.
TIRDCRYP-CMCB	input/ output	A pointer to a structure containing the following items:

Name	I/O	Description		
		DATA-BUFFER-PTR	input/ output	On input, a pointer to the starting location of the encrypted View Data and Client Security sections within the CFB work buffer. On output, a pointer to the starting location of the area that now contains the unencrypted version of the input data. The length of this decrypted data cannot exceed DECRYPTION-MAX-SIZE.
		BUFFER-SIZE	input/ output	On input, BUFFER-SIZE is the current buffer size (in bytes) of the encrypted data. On output, BUFFER-SIZE should be updated by this exit to contain the length of the decrypted data. The length of the decrypted data cannot exceed DECRYPTION-MAX-SIZE.
		DECRYPTION- MAX-SIZE	input	A field that contains the maximum available buffer space (in bytes) that the decrypted data can occupy.

Name	I/O	Description		
		TIRDCRYP-RETURN-CODE	output	A 2-byte character field returning the result of the decryption attempt. The following values are supported: DECRYPTION-USED—defined as SPACES DECRYPTION-SIZE-EXCEEDED-MAX—defined as 01 DECRYPTION-NOT-USED—defined as 02 DECRYPTION-APPLICATION-ERROR—defined as 03
		FAILURE-MSG	output	An 80-byte character field, to be populated by this exit, to describe the failure with a message of choice. This failure message will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of DECRYPTION-APPLICATION-ERROR.

Return Code

See TIRDCRYP-RETURN-CODE in the preceding Arguments section.

Default Processing

The decryption user exit is only invoked if an encrypted buffer is received. By default, this user exit sets the return code of decryption-not-used, indicating an error. If the client sends an encrypted buffer, this exit needs to be modified to properly decrypt the data.

Customizing the Exit

This exit is invoked when the cooperative buffer received from the client contains encrypted data. Decrypt the data pointed to DATA-BUFFER-PTR for the length BUFFER-SIZE bytes using the appropriate decryption algorithm. Ensure the DATA-BUFFER-PTR points to the location of the decrypted data and that the BUFFER-SIZE is updated with the correct data length. The maximum size of the decrypted data cannot exceed DECRYPTION-MAX-SIZE. Set the return code to DECRYPTION-USED.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

WRSECENCRYPT

TIRNCRYX—Server Encryption Exit

The CA Gen Server Encryption Exit is used by z/OS servers.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRNCRYX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                      PIC X.
01  RUNTIME-PARM2                      PIC X.
01  TIRNCRYP-CMCB.
    03  DATA-BUFFER-PTR                        POINTER.
    03  BUFFER-SIZE                            PIC 9(09) COMP.
    03  ENCRYPTION-MAX-SIZE                    PIC 9(09) COMP.
    03  TIRNCRYP-TRANCODE                PIC X(08).
    03  CLIENT-USERID                          PIC X(64).
    03  NEXT-LOCATION-PTR                  POINTER.
    03  TIRNCRYP-RETURN-CODE           PIC X(02).
    03  FAILURE-MSG                            PIC X(80).
```

Purpose

TIRNCRYX is called by the Server Runtime to allow encryption of the cooperative buffer before the response is sent to the client. The user is responsible for encrypting the data pointed to by DATA-BUFFER-PTR for the length BUFFER-SIZE bytes. The maximum size of the encrypted data cannot exceed ENCRYPTION-MAX-SIZE.

Note: Encryption/decryption of the request and response cooperative buffers are mutually exclusive—encryption/decryption of the request buffer does not necessitate that the same be done to the response buffer and vice-versa.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	Under CICS, this is CICS EXEC Interface Block - DFHEIBLK.
RUNTIME-PARM2	input	Under CICS, this is DFHCOMMAREA.
TIRNCRYP-CMCB	input/output	A pointer to a structure containing the following items:
DATA-BUFFER-PTR	input/ output	On input, a pointer to the starting location of the View Data and Client Security sections within the CFB work buffer. On output, a pointer to the starting location of the area that now contains the encrypted version of the input data. The length of this encrypted data cannot exceed ENCRYPTION-MAX-SIZE.
BUFFER-SIZE	input/ output	On input, BUFFER-SIZE is the current buffer size (in bytes) of the data to be encrypted. On output, BUFFER-SIZE should be updated by this exit to contain the length of the encrypted data. The length of the encrypted data cannot exceed ENCRYPTION-MAX-SIZE.
ENCRYPTION-MAX-SIZE	input	A field that contains the maximum available buffer space (in bytes) that the encrypted data can occupy.
TIRNCRYP-TRANCODE	input	A field that contains the trancode identifying the target server.

Name	I/O	Description
CLIENT-USERID	input	A field containing the userid originally sent by the client.
NEXT-LOCATION-PTR	input	Not used.
TIRNCRYP-RETURN-CODE	output	A 2-byte character field returning the result of the encryption attempt. The following values are supported: ENCRYPTION-USED—defined as SPACES ENCRYPTION-SIZE-EXCEEDED-MAX—defined as 01 ENCRYPTION-NOT-USED—defined as 02 ENCRYPTION-APPLICATION-ERROR—defined as 03
FAILURE-MSG	output	An 80-byte character field, to be populated by this exit, to describe the failure with a message of choice. This failure message will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of ENCRYPTION-APPLICATION-ERROR.

Return Code

See TIRNCRYP-RETURN-CODE in the preceding Arguments section.

Default Processing

Encryption of the data buffer is not attempted. The TIRNCRYP-RETURN-CODE is set to ENCRYPTION-NOT-USED.

Customizing the Exit

This exit is always invoked by the server runtime before the response is sent to the client. The exit must update TIRNCRYP-RETURN-CODE field to indicate whether the data is encrypted. The data that can be encrypted buffer pointed to by DATA-BUFFER-PTR with length BUFFER-SIZE bytes. Ensure that DATA-BUFFER-PTR points to the location of the encrypted data and that BUFFER-SIZE is updated with the correct data length. The maximum size of the encrypted data cannot exceed ENCRYPTION-MAX-SIZE

Building on z/OS

For information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

WRSECDECRYPT

TIRELOGX—Server Error Logging and Error Token Creation Exit

The CA Gen Server Error Logging and Error Token Creation Exit is used by z/OS servers.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRELOGX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                          PIC X.
01  RUNTIME-PARM2                          PIC X.
01  ELOG-EXIT-PARM-LIST.
03  ELOG-FAIL-TYPE                         PIC X.
03  ELOG-SQLCA.
    05  ELOG-SQLCA-ADR                     PIC S9(9) COMP.
    05  ELOG-SQLCA-PTR          REDEFINES ELOG-SQLCA-ADR POINTER.
03  ELOG-GLOBDATA.
    05  ELOG-GLOBDATA-ADR                  PIC S9(9) COMP.
    05  ELOG-GLOBDATA-PTR                  REDEFINES ELOG-GLOBDATA-ADR POINTER.
03  ELOG-ERROR.
    05  ELOG-NUMBER-OF-LINES               PIC S9(9) COMP.
    05  ELOG-ERROR-TEXT                    PIC X(80) OCCURS 31.
03  ELOG-ERROR-TOKEN.
        05  ELOG-ERROR-TOKEN-ADR           PIC X(4).
        05  ELOG-ERROR-TOKEN-PTR           REDEFINES ELOG-ERROR-TOKEN-ADR POINTER.
```

Purpose

The server error handling routine calls TIRELOGX before the error response is returned to the client to let the server log the error locally or/and create an error token to be sent to the client.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	Under CICS, this is CICS EXEC Interface Block - DFHEIBLK.

RUNTIME-PARM2	input	Under CICS, this is DFHCOMMAREA.		
LOG-EXIT-PARM-LIST	input/ output	A structure containing the following items:		
ELOG-FAIL-TYPE	input	The type of error.		
ELOG-SQLCA	input	A pointer to SQLCA.		
ELOG-GLOBDATA	input	A pointer to GLOBDATA.		
ELOG-ERROR	input	A structure with the following items:		
		ELOG-NUMBER-OF-LINES	input	The number of error messages used.
		ELOG-ERROR-TEXT	input/ output	A table of 80-byte error messages.
ELOG-ERROR-TOKEN	output	A pointer to the error token, maximum length 4096 bytes.		

Return Code

No explicit return code value. This exit can optionally return data contained in area pointed to by ELOG-ERROR-TOKEN-PTR.

Default Processing

Return without any action.

Customizing the Exit

This exit can be used to enable the server to log the error and create an error token to be sent to the client. Either of these actions can be done separate from each other.

Server Error Logging—The error information provided in ELOG-ERROR can be used to log a message on the server platform.

Error Token Creation—The error information provided in ELOG-ERROR can be used to create a text message. A pointer to this message must be provided in ELOG-ERROR-TOKEN. The message is translated to the code page used by the client and can be processed by the Client/Server Flow Server Failure Exit or Client/Server Asynchronous Flow Server Failure Exit to customize how the error is handled.

Building on z/OS

For more information about installing the exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- WRSRVRERROR
- WRASYNCSRVRERROR

TIRALLOX—Server-to-Server Allocate Conversation Exit

z/OS servers use the CA Gen Server-to-Server Allocate Conversation Exit.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRALLOX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01 RUNTIME-PARM1                    PIC X.
01 RUNTIME-PARM2                    PIC X.
01 GLOBDATA                                 size 3645 bytes
01 SERVER-CONNECTION-DATA           PIC X(256).
01 ASYNCH-FLAG                      PIC X(01).
01 PSTEP-TRANCODE                   PIC X(08).
01 PSTEP-LOADMOD                    PIC X(08).
01 ALLOCATE-PARTNER-NAME            PIC X(08).
01 ALLOCATE-PROFILE-NAME            PIC X(08).
01 WAIT-TRAN-TIMEOUT                PIC S9(7)  COMP-3.
01 MAIN-OR-AUX-STORAGE              PIC X(01).
01 LOCAL-RMT-FLAG                   PIC X(01).
01 TIRALLOX-RETURN                  PIC X(02).
01 FAIL-MESSAGE                     PIC X(75).
```

Purpose

For a Server-to-Server flow, CA Gen's server runtime invokes the TIRALLOX user exit to set the type of Server-to-Server flow and obtain the relevant parameters required by each type. There are two distinct types: local or remote.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is CICS EXEC Interface Block - DFHEIBLK.
RUNTIME-PARM2	input	This is DFHCOMMAREA.
GLOBDATA	input	Global data, used internally.
SERVER-CONNECTION-DATA	input	Name of target host where the target server is to execute as specified by the TIRPROUX exit.
ASYNCH-FLAG	input	Flag indicating Asynchronous request. Not used.
PSTEP-TRANCODE	Input	Target server transaction ID.
PSTEP-LOADMOD	Input	Target server load module name.
ALLOCATE-PARTNER-NAME	output	Partner name to be used in the SNA allocate command.
ALLOCATE-PROFILE-NAME	output	Profile name to be used in the SNA allocate command.
WAIT-TRAN-TIMEOUT	output	A packed field to specify the amount of time (HHMMSS) after which the CICS Server-to-Server flow is considered non-responsive.
MAIN-OR-AUX-STORAGE	output	A flag indicating where to write the cooperative buffer for local CICS Server-to-Server requests. Values can be M for Main Storage or A for Auxiliary Storage.
LOCAL-RMT-FLAG	output	Flag to indicate whether this is for a local or remote CICS Server-to-Server request.
TIRALLOX-RETURN	output	A 2-byte character field indicating result of the exit. The following values are supported: SPACES—successful NON-SPACES—failure

Name	I/O	Description
FAIL-MESSAGE	output	A 75-byte character field that if populated will be returned to the client.

Return Code

See TIRALLOX-RETURN in the preceding Arguments section.

Default Processing

For CICS applications the default processing of this exit is to use, the Server Connection Data passed from the TIRPROUX exit to populate the ALLOCATE-PARTNER-NAME while the ALLOCATE-PROFILE-NAME is set to spaces. The TIRSYSIX exit is called to obtain the name of the system the current application is executing in and if that matches the ALLOCATE-PARTNER-NAME a Local request is assumed, otherwise a Server-to-Server Remote request is issued. The WAIT-TRAN-TIMEOUT is set to 1 minute and CICS Main Storage is selected as the location of the TSQ used for the cooperative buffer.

Customizing the Exit

This exit is used by the server runtime to enable changing the parameters required to issue the Server-to-Server request. TIRALLOX-RETURN must be set to spaces to indicate success or non-space to indicate failure. In case of failure, the FAIL-MESSAGE can be populated with a message that will be returned to the client.

The CICS implementation differentiates between a Local and Remote requests.

A Local request means both servers will execute in the same CICS region. The method used for this implementation is to call user exit TIRCQNAM, issue a CICS Start command to start the target server transaction passing it the name of a TSQ containing the cooperative buffer, issue a CICS Post ECB with an interval of WAIT-TRAN-TIMEOUT and a CICS Wait ECB. User exit TIRCQNAM provides a customized name for the TSQ used in the CICS Start command and located as specified in MAIN-OR-AUX-STORAGE parameter.

Modify the MAIN-OR-AUX-STORAGE or the WAIT-TRAN-TIMEOUT as required by the application. Note that the WAIT-TRAN-TIMEOUT value should be large enough to enable all activity of the Server-to-Server flow to complete.

A Remote request means each server executes in a different CICS region using an APPC (LU6.2) connection between the two CICS. A CICS Allocate Sysid command using the ALLOCATE-PARTNER-NAME as the Sysid is used to allocate the conversation followed by a CICS Connect Process and CICS Send using the Convid returned by the Allocate. The ALLOCATE-PROFILE-NAME becomes DFHCICSA if left blank. Modify the ALLOCATE-PARTNER-NAME or ALLOCATE-PROFILE-NAME as required by the application. The CICS APPC definitions need to be in place, the Profile must include the name of the VTAM Logmode Table as the MODENAME and the Connections must be ACQUIRED prior to the CICS Allocate being issued.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRPROUX
- TIRPTOKX
- TIRSYSIX
- TIRCQNAM

TIRPTOKX—Server-to-Server Security Token CA Generation Exit

z/OS servers use the CA Gen Server-to-Server Security Token CA Generation Exit.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRPTOKX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01  RUNTIME-PARM1                          PIC X(01).
01  RUNTIME-PARM2                          PIC X(01).
01  GLOBDATA                                      size 3645 bytes.
01  TEMP-BUFFER                            PIC X(32768).
01  USE-SECR-FLAG                          PIC S9(04) COMP.
01  TOKEN-SIZE                             PIC S9(09) COMP.
01  FAIL-MESSAGE                           PIC X(75).
```

Purpose

Server-to-Server Runtime calls TIRPTOKX to enable the generation of a Security Token to be placed in the security section of the cooperative buffer to be sent to the target server.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	Under CICS, this is CICS EXEC Interface Block -DFHEIBLK.
RUNTIME-PARM2	input	Under CICS, this is DFHCOMMAREA.
GLOBDATA	input	Global data, used internally.
TEMP-BUFFER	input	Buffer for storing Token
USE-SECR-FLAG	output	Security Flag. The following values are supported: -1 indicating SECURITY NOT USED +0 indicating SECURITY USED, OK +1 indicating SECURITY ERROR
TOKEN-SIZE	output	Size of generated Token
FAIL-MESSAGE	output	A 75-byte character field that if populated will be returned to the client.

Return Code

See USE-SECR-FLAG in the preceding Arguments section.

Default Processing

The TIRPTOKX exit sets the USE-SECR-FLAG to NOT USED (-1) and the TOKEN-SIZE to zero.

Customizing the Exit

This exit can be customized to return a security-token in TEMP-BUFFER with its length in TOKEN-SIZE. If modified the USE-SECR-FLAG must also be updated to indicate SECURITY USED, OK, or SECURITY ERROR as applicable. The runtime will take the TOKEN-SIZE and the generated Token together with the current application's system attributes CLIENT-USERID and CLIENT-PASSWORD and create a Security Offset in the cooperative buffer for the Server-to-Server request.

Building on z/OS

For more information about installing the exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

TIRALLOX

TIRCSGNX—Server TCP/IP Signon Exit

The CA Gen Server Signon Exit is used by z/OS Servers that use TCP/IP Direct Connect Option for CICS. The server calls this exit, but not TICONMGR.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library, in member TIRCSGNX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01 RUNTIME-PARM1                PIC X(01).
01 RUNTIME-PARM2                PIC X(01).
01 TIRCSGN-RETURN                PIC X(02).
01 SGN-USERID           PIC X(08).
```

Purpose

TCP/IP servers started with a terminal ID parameter are intended to use this exit, but not the userid parameter to obtain the client userid. This is for applications that have been designed to have special knowledge of where to obtain the required information.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	CICS DFHEIBLK, automatically included if translated.
RUNTIME-PARM2	input	CICS DFHCOMMAREA, automatically included if translated.

Name	I/O	Description
TIRCSGN-RETURN	output	A 2-byte character field indicating result of the exit. The following values are supported: SPACES—Successful NON-SPACES—Failure
SGN-USERID	input	The userid as provided by the client application.

Return Code

See TIRCSGN-RETURN in the preceding Arguments section.

Default Processing

This exit moves spaces to TIRCSGN-RETURN and returns.

Customizing the Exit

TCP/IP Direct Connect CICS servers run as CICS non-terminal tasks started using the CICS START command with the USERID option. This USERID option specifies the userid sent by the client application. However, TCP/IP Direct Connect CICS servers can be started as CICS terminal tasks by using CICS START command with the TERMID option when a value for a terminal is provided in the Server Termid parameter of the CICS Sockets Server Listener Exit TIRSLEXT. The userid and termid are mutually exclusive options of the CICS START command and these applications run under the userid provided by CICS to TISRVLIS Listener.

This exit provides the opportunity for servers designed with knowledge of where to obtain the client security data to be able to obtain and use this data.

The exit is intended to use the SGN-USERID. The server will not pick up any changes to its value.

Building on z/OS

It is strongly recommended that this exit not be modified so the TCP/IP Servers execute without a terminal ID but with a userid.

For more information about installing the exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None

TIRPROUX—Server-to-Server Routing Exit

z/OS servers use the CA Gen Server-to-Server Routing Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library, in member TIRCROUX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRCROUX is as follows:

```
01  GLOBDATA                                          structure size  3645
bytes.
01  PSTEP-NAME                                  PIC X(32).
01  PSTEP-TRANCODE                              PIC X(08).
01  PSTEP-LOADMOD                               PIC X(08).
01  ROUTR-OUT-TRANCODE                          PIC X(08).
01  ROUTR-OUT-LOADMOD                           PIC X(08).
01  ROUTR-OUT-SVR-CONNECT-DATA                  PIC X(256).
01  ROUTR-OUT-ASYNCH-FLAG                       PIC X(01).
01  ROUTR-OUT-RET-CODE                          PIC X(02).
01  FAIL-MESSAGE                                PIC X(75).
```

This exit contains CICS API calls that require it to be processed by the Translator. The Translator automatically includes data structures for DFHEIBLK and DFHCOMMAREA in the place of RUNTIME-PARM1 and RUNTIME-PARM2 thus RUNTIME-PARMx are not specified.

Purpose

Server-to-Server Runtime calls TIRPROUX to let the target server node name, trancode and load module name be specified.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	Under CICS, this is CICS EXEC Interface Block DFHEIBLK.
RUNTIME-PARM2	input	Under CICS, this is DFHCOMMAREA.
GLOBDATA	input	Global data, used internally.

Name	I/O	Description
PSTEP-NAME	input	Target server procedure step name.
PSTEP-TRANCODE	input	Target server transaction ID.
PSTEP-LOADMOD	input	Target server load module name.
ROUTR-OUT-TRANCODE	output	Modified target server transaction ID.
ROUTR-OUT-LOADMOD	output	Modified target server load module name.
ROUTR-OUT-SVR-CONNECT-DATA	output	Name of target node where target server is to execute.
ROUTR-OUT-ASYNCH-FLAG	output	Flag indicating Asynchronous request. Not used.
ROUTR-OUT-RET-CODE	output	A 2-byte character field indicating whether the exit was successful. The following values are supported: SPACES—successful NON-SPACES—failure
FAIL-MESSAGE	output	A 75-byte character field that if populated will be returned to the client.

Return Code

See TIRSECV-RETURN-CODE in the preceding Arguments section.

Default Processing

The CICS implementation calls TIRSYSIX user exit to obtain the Sysid and returns it as the ROUTR-OUT-SVR-CONNECT-DATA (server node name) and spaces as the ROUTR-OUT-TRANCODE and ROUTR-OUT-LOADMOD names.

Customizing the Exit

CICS

The CICS implementation of this exit can be modified to update the ROUTR-OUT-SVR-CONNECT-DATA, the ROUTR-OUT-TRANCODE, and ROUTR-OUT-LOADMOD as required.

IMS

The IMS implementation of this exit can be modified to update the ROUTR-OUT-SVR-CONNECT-DATA, the ROUTR-OUT-TRANCODE, and ROUTR-OUT-LOADMOD as required. The ROUTR-OUT-SVR-CONNECT-DATA is used by the TIRALLOC exit as the SERVER-CONNECTION-DATA.

Building on z/OS

For more information about installing the exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRALLOC
- TIRPTOKX
- TIRSYSIX

TIRSIPEX—CICS Sockets Server Exit

The CA Gen CICS Sockets Server Exit is used by the z/OS CICS servers started by the CICS Socket Server Listener program TISRVLIS, the CA Gen TCP/IP implementation.

The exit is used to allow customization of various variables used by the server when processing a request.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library, in member TIRSIPEX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The parameter list used by TIRSIPEX is as follows:

```
01  RUNTIME-PARM1                  PIC X.
01  RUNTIME-PARM2                  PIC X.
01  SIP-TS-PREFIX                          PIC X(03).
01  DEST-ERR                               PIC X(04).
01  DEST-INFO                      PIC X(04).
01  SELECT-TIMEOUT-SECS            PIC 9(08) BINARY.
01  SELECT-TIMEOUT-MICROSEC        PIC 9(08) BINARY.
01  ERROR-IP-FLAG                  PIC X(1).
01  TIRSIPEX-RC                    PIC X(02).
```

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
RUNTIME-PARM1	input/ output	In CICS, this becomes DFHEIBLK, which is automatically included if translated
RUNTIME-PARM2	input/ output	In CICS, this becomes DFHCOMMAREA which is automatically included if translated
SIP-TS-PREFIX	input/ output	Prefix to be used for the temporary storage queue (TSQ) to hold the Socket Descriptor. This should be a local TSQ.
DEST-ERR	input/ output	Name of transient data queue (TDQ) used to report error messages. Messages will be suppressed if the queue does not exist.
DEST-INFO	input/ output	Name of transient data queue (TDQ) used to report informational messages. Messages will be suppressed if the queue does not exist.
SELECT-TIMEOUT- SECS	input/ output	Number of seconds the Sockets API SELECT call will wait to timeout.
SELECT-TIMEOUT- MICROSEC	input/ output	Number of micro seconds the Sockets API SELECT call will wait to timeout.

Name	I/O	Description
ERROR-IP-FLAG	output	A 1-byte flag that tells the server whether to add the IP address to some error messages: N – do not add Y - add
TIRSIPEX-RC	output	A two-byte character field indicating result of the exit. The following values are supported: SPACES – Successful NON-SPACES – Failure

Return Code

See TIRSIPEX-RC in the preceding Arguments section.

Default Processing

The default processing of this exit sets SIP-TS-PREFIX to SIP, DEST-ERR to CSSL, DEST-INFO to TISL, SELECT-TIMEOUT-SECS to zero, SELECT-TIMEOUT-MICROSEC to 100 microseconds, ERROR-IP-FLAG to N, and TIRSIPEX-RC to spaces.

Customizing the Exit

This exit does not use CICS commands so it does not need to be translated for CICS and it specifies RUNTIME-PARM1 and RUNTIME-PARM2 in both the Linkage Section and the Procedure Division statement.

The SIP-TS-PREFIX parameter is used in conjunction with the CICS taskid to make up the name of a unique TSQ used by the server. This TSQ should be local and is required during the execution of each server. If changed, this parameter must be unique per CICS region.

The DEST-ERR parameter provides the name of a transient data queue (TDQ) where error messages are written to. If changed, ensure this TDQ exists to obtain these required messages.

The DEST-INFO parameter provides the name of a transient data queue (TDQ) where informational messages are written to. If this TDQ does not exist the informational messages will be suppressed.

The SELECT-TIMEOUT-SECS and SELECT-TIMEOUT-MICROSEC parameters combine to become the TIMEOUT parameter of the Socket SELECT call. This SELECT call is used to determine if the socket contains data to be retrieved before doing a Socket RECV call or Socket READ call.

The ERROR-IP-FLAG parameter is used to tell the server runtime whether to add the IP address to the some socket and CICS error messages. The default is to not add the IP address.

Modify the parameters as required by the application.

Building on z/OS

For information about installing the exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRSLEXT
- TIRSLTMX

TIRMQPX—MQ SERIES Put Function Exit

The CA Gen MQ Series Server Exit is used by the z/OS CICS and IMS servers started by the MQ Series. The exit is used to allow customization of syncpoint variable used by the server in the processing of the request.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library, in member TIRMQPX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRMQPX is as follows:

```
01  RUNTIME-PARM1                          PIC X.
01  RUNTIME-PARM2                          PIC X.
01  TRANCODE                               PIC X(08).
01  SYNCPOINT-FLAG                         PIC X(1).
```

Purpose

This exit can be used to customize the server runtime MQPUT function to do a syncpoint instead of no syncpoint. The default processing is to do no syncpoint.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
RUNTIME-PARM1	input/ output	In CICS, this becomes DFHEIBLK, which is automatically included if translated
RUNTIME-PARM2	input/ output	In CICS, this becomes DFHCOMMAREA which is automatically included if translated
TRANCODE	input/ output	The server trancode
SYNCPOINT-FLAG	input/ output	Flag set to either Y or N. Default is N

Return Code

There is no explicit return code value provided by the TIRMQPX exit.

Default Processing

The default processing of this exit sets SYNCPOINT-FLAG to N.

Customizing the Exit

This exit does not use CICS commands so it does not need to be translated for CICS and it specifies RUNTIME-PARM1 and RUNTIME-PARM2 in both the Linkage Section and the Procedure Division statement.

The TRANCODE parameter can be used to decide if the MQPUT function should specify SYNCPOINT instead of NO-SYNCPOINT.

The SYNCPOINT-FLAG parameter should be set to Y if MQPUT should use SYNCPOINT.

Modify the parameters as required by the application.

Building on z/OS

You will need to use the MKORUNX jcl in the CEHBSAMP library.

Related User Exits

None

z/OS Server User Exits—IMS

TIRTIARX—DB2 Message Exit

z/OS Server Managers use the CA Gen DB2 Message Exit.

Source Code

The source code for the version of the exit used by IMS application is in CA Gen CEHBSAMP library, in member TIRITIAX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRTIARX is as follows:

```
01  RUNTIME-PARM1                PIC X.
01  RUNTIME-PARM2                PIC X.
01  TIRFAIL-SQLCA                PIC X.
01  TIRTIAR-ERRORS               PIC X.
01  TIRTIAR-TEXT-LEN             PIC X.
01  TIRTIAR-WORKAREA            PIC X.
01  GLOBDATA                    structure size 3645 bytes.
```

Purpose

The DB2 Message Exit is used by all applications targeting DB2 database on z/OS. The TIRFAIL subroutine of the Dialog Manager calls the DB2 Message exit, TIRTIARX, whenever an unrecoverable DB2 failure occurs. TIRTIARX then calls the subroutine DSNTIAR to convert the SQL code into text. The messages returned by DSNTIAR are then merged with the runtime error messages.

TIRTIARX can be customized to statically link DSNTIAR with the executable load module rather than dynamically linking it.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is IO-PCB automatically included if translated.

Name	I/O	Description
RUNTIME-PARM2	input	This is ALT-IO-PCB automatically included if translated.
TIRFAIL-SQLCA	input	SQLCA
TIRTIAR-ERRORS	input/output	Error message lines.
TIRTIAR-TEXT-LEN	input	Length of one error message line.
TIRTIAR-WORKAREA	input	Workarea
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Behavior

As provided by CA Gen, the default exit dynamically calls DSNTIAR and is compatible with prior releases. However the sample code also contains examples of how to call DSNTIAR or DSNTIAC statically.

The call to TIRTIARX is made when TIRFAIL is building the table of messages and occurs prior to calling the default termination exit. For more information, see the Online Termination Exit and Batch Termination Exit.

Customizing the DB2 Message Exit

Copy the default exit to one of your own libraries. The member name for IMS is TIRITIAX. The default exit includes example code for the two possible combinations of calls. There are dynamic and static calls of DSNTIAR. Simply comment out the default call and remove the comments from the one you want to use.

When you have completed your modifications, install your exit.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRALLOX—Server-to-Server Allocate Conversation Exit

z/OS servers use the CA Gen Server-to-Server Allocate Conversation Exit.

Source Code

The sample source for this exit can be found in CA Gen CEHBSAMP library, in member TIRALLOX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by this exit is as follows:

```
01 RUNTIME-PARM1                    PIC X.
01 RUNTIME-PARM2                    PIC X.
01 GLOBDATA                                 size 3645 bytes
01 SERVER-CONNECTION-DATA           PIC X(256).
01 ASYNCH-FLAG                      PIC X(01).
01 PSTEP-TRANCODE                   PIC X(08).
01 PSTEP-LOADMOD                    PIC X(08).
01 ALLOCATE-PARTNER-NAME            PIC X(08).
01 ALLOCATE-PROFILE-NAME            PIC X(08).
01 WAIT-TRAN-TIMEOUT                PIC S9(7)  COMP-3.
01 MAIN-OR-AUX-STORAGE              PIC X(01).
01 LOCAL-RMT-FLAG                   PIC X(01).
01 TIRALLOX-RETURN                  PIC X(02).
01 FAIL-MESSAGE                     PIC X(75).
```

Purpose

For a Server-to-Server flow, CA Gen's server runtime invokes the TIRALLOX user exit to set the type of Server-to-Server flow and obtain the relevant parameters required by each type. There are two distinct types: local or remote.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is the I/O PCB.
RUNTIME-PARM2	input	This is Alternate I/O PCB.
GLOBDATA	input	Global data, used internally.
SERVER-CONNECTION-DATA	input	Name of target host where the target server is to execute as specified by the TIRPROUX exit.

Name	I/O	Description
ASYNCH-FLAG	input	Flag indicating Asynchronous request. Not used.
PSTEP-TRANCODE	Input	Target server transaction ID.
PSTEP-LOADMOD	Input	Target server load module name.
ALLOCATE-PARTNER-NAME	output	Partner name to be used in the SNA allocate command.
ALLOCATE-PROFILE-NAME	output	Profile name to be used in the SNA allocate command.
WAIT-TRAN-TIMEOUT	output	A packed field to specify the amount of time (HHMMSS) after which the Server-to-Server flow is considered non-responsive.
MAIN-OR-AUX-STORAGE	output	A flag indicating where to write the cooperative buffer for local Server-to-Server requests. Values can be M for Main Storage or A for Auxiliary Storage.
LOCAL-RMT-FLAG	output	Flag to indicate whether this is for a local or remote Server-to-Server request.
TIRALLOX-RETURN	output	A 2-byte character field indicating result of the exit. The following values are supported: SPACES—successful NON-SPACES—failure
FAIL-MESSAGE	output	A 75-byte character field that if populated will be returned to the client.

Return Code

See TIRALLOX-RETURN in the preceding Arguments section.

Default Processing

For IMS applications the default processing of this exit is to use, the Server Connection Data passed from the TIRPROUX exit to populate the ALLOCATE-PARTNER-NAME. This is the target server trancode.

All the other parameters are ignored.

Customizing the Exit

This exit is used by the server runtime to enable changing the parameters required to issue the Server-to-Server request. TIRALLOX-RETURN must be set to spaces to indicate success or non-space to indicate failure. In case of failure, the FAIL-MESSAGE can be populated with a message that will be returned to the client.

The IMS implementation does not use the LOCAL-RMT_FLAG to identify the request as local or remote. Instead, it uses the CPI-C services of APPC/IMS and APPC/MVS as the communication mechanism for Server-to-Server requests and obtains the required information from the CPI-C Side Information File. This file is a VSAM KSDS data set with the Server Trancode name as key to the Symbolic Destination Entry. The Symbolic Destination Entry contains the PARTNER_LU name (the server node name), MODENAME (VTAM Logmode Table) and the TPNAME (trancode) required by APPC. If the PARTNER_LU is blank, the request is considered local, and APPC targets the same system as the current application, otherwise it uses the PARTNER_LU as the server node name for the Remote request. Modify the ALLOCATE-PARTNER-NAME as required by the application and configure the CPI-C Side Info File accordingly. IBM provides utility ATBSDFMU to customize the Side Info File.

Note: For more information about APPC/MVS, see the IBM Guide.

Building on z/OS

For more information about installing this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRPROUX
- TIRPTOKX
- TIRSYSIX
- TIRCQNAM

TIRPROUX—Server-to-Server Routing Exit

z/OS servers use the CA Gen Server-to-Server Routing Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library, in member TIRIROUX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRIROUX is as follows:

```
01  RUNTIME-PARM1                          PIC X.
01  RUNTIME-PARM2                          PIC X.
01  GLOBDATA                                       size 3645 bytes.
01  PSTEP-NAME                             PIC X(32).
01  PSTEP-TRANCODE                         PIC X(08).
01  PSTEP-LOADMOD                          PIC X(08).
01  ROUTR-OUT-TRANCODE                     PIC X(08).
01  ROUTR-OUT-LOADMOD                      PIC X(08).
01  ROUTR-OUT-SVR-CONNECT-DATA             PIC X(256).
01  ROUTR-OUT-ASYNCH-FLAG                  PIC X(01).
01  ROUTR-OUT-RET-CODE                     PIC X(02).
01  FAIL-MESSAGE                           PIC X(75).
```

Purpose

Server-to-Server Runtime calls TIRPROUX to let the target server node name, trancode and load module name be specified.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is the I/O PCB.
RUNTIME-PARM2	input	This is Alternate I/O PCB.
GLOBDATA	input	Global data, used internally.
PSTEP-NAME	input	Target server procedure step name.
PSTEP-TRANCODE	input	Target server transaction ID.
PSTEP-LOADMOD	input	Target server load module name.
ROUTR-OUT-TRANCODE	output	Modified target server transaction ID.
ROUTR-OUT-LOADMOD	output	Modified target server load module name.
ROUTR-OUT-SVR-CONNECT-DATA	output	Name of target node where target server is to execute.

Name	I/O	Description
ROUTR-OUT-ASYNCH-FLAG	output	Flag indicating Asynchronous request. Not used.
ROUTR-OUT-RET-CODE	output	A 2-byte character field indicating whether the exit was successful. The following values are supported: SPACES—successful NON-SPACES—failure
FAIL-MESSAGE	output	A 75-byte character field that if populated will be returned to the client.

Return Code

See TIRSECV-RETURN-CODE in the preceding Arguments section.

Default Processing

The IMS implementation returns the Server Trancode as the ROUTR-OUT-SVR-CONNECT-DATA and spaces as the ROUTR-OUT-TRANCODE and ROUTR-OUT-LOADMOD names.

Customizing the Exit

The IMS implementation of this exit can be modified to update the ROUTR-OUT-SVR-CONNECT-DATA, the ROUTR-OUT-TRANCODE, and ROUTR-OUT-LOADMOD as required. The ROUTR-OUT-SVR-CONNECT-DATA is used by the TIRALLOC exit as the SERVER-CONNECTION-DATA.

Building on z/OS

For more information about installing the exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

The following are related user exits:

- TIRALLOC
- TIRPTOKX
- TIRSYSIX

z/OS Batch User Exits

TIRTIARX—DB2 Message Exit

z/OS Batch Managers use the CA Gen DB2 Message Exit.

Source Code

The source code for the version of the exit used by Batch application is in CA Gen CEHBSAMP library, in member TIRITIAX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRTIARX is as follows:

```
01  RUNTIME-PARM1                PIC X.
01  RUNTIME-PARM2                PIC X.
01  TIRFAIL-SQLCA                PIC X.
01  TIRTIAR-ERRORS               PIC X.
01  TIRTIAR-TEXT-LEN             PIC X.
01  TIRTIAR-WORKAREA            PIC X.
01  GLOBDATA                    structure size 3645 bytes.
```

Purpose

The DB2 Message Exit is used by all applications targeting DB2 database on z/OS. The TIRFAIL subroutine of the Dialog Manager calls the DB2 Message exit, TIRTIARX, whenever an unrecoverable DB2 failure occurs. TIRTIARX then calls the subroutine DSNTIAR to convert the SQL code into text. The messages returned by DSNTIAR are then merged with the runtime error messages.

TIRTIARX can be customized to statically link DSNTIAR with the executable load module rather than dynamically linking it.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is IO-PCB automatically included if translated.

Name	I/O	Description
RUNTIME-PARM2	input	This is ALT-IO-PCB automatically included if translated.
TIRFAIL-SQLCA	input	SQLCA
TIRTIAR-ERRORS	input/output	Error message lines.
TIRTIAR-TEXT-LEN	input	Length of one error message line.
TIRTIAR-WORKAREA	input	Workarea.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Behavior

As provided by CA Gen, the default exit dynamically calls DSNTIAR and is compatible with prior releases. However the sample code also contains examples of how to call DSNTIAR or DSNTIAC statically.

The call to TIRTIARX is made when TIRFAIL is building the table of messages and occurs prior to calling the default termination exit. For more information, see the Online Termination Exit and Batch Termination Exit.

Customizing the DB2 Message Exit

Copy the default exit to one of your own libraries. The member name for batch is TIRITIAX. The default exit includes example code for the two possible combinations of calls. There are dynamic and static calls of DSNTIAR. Simply comment out the default call and remove the comments from the one you want to use.

When you have completed your modifications, install your exit.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

TIRBRTRX—Default Retry Limit Exit

z/OS Batch Managers for Batch applications use the CA Gen Default Retry Limit Exit.

Source Code

The source code for this exit is in CA Gen CEHBSAMP library, in member TIRBRTRX. The sample exit provided is written in COBOL and uses standard OS Linkage.

The Linkage Parameter list used by TIRBRTRX is as follows:

```
01  RETRY-TIMES         PIC S9(4) COMP.
```

Purpose

This exit is called at the beginning of a CA Gen Batch application to enable the defined default value for the TRANSACTION RETRY LIMIT system attribute to be modified.

TRANSACTION RETRY LIMIT will be initialized to this value at the beginning of each new execution. This value may subsequently be modified by a SET TRANSACTION RETRY LIMIT statement in an action diagram.

TRANSACTION RETRY LIMIT is used to specify the maximum number of times a transaction is to be retried when one of the following events occurs:

- A RETRY TRANSACTION action diagram statement executes.

- A deadlock or timeout occurs trying to access a database, and no WHEN DATABASE DEADLOCK OR TIMEOUT statement was provided for that entity action statement. (This is not applicable for z/OS transactions running under IMS or IEFAE. An application running under IMS that encounters a deadlock or timeout will be terminated immediately by IMS itself, even if it has a WHEN DATABASE DEADLOCK OR TIMEOUT statement provided.)

In either of these cases, any uncommitted database updates will be rolled back, and an attempt will then be made to execute the application again. Once the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, no more retries can occur, and the application will fail with a runtime error if the last retry attempt was unsuccessful.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
ULTIMATE-RETRY-LIMIT	input/ output	Absolute number of retries allowed in a batch appliation.

Return Code

No explicit return code is set by the user exit.

This exit is called at the beginning of a CA Gen CICS blockmode application to enable the defined default value for the TRANSACTION RETRY LIMIT system attribute to be modified. The TRANSACTION RETRY LIMIT will be initialized to this value at the beginning of each new transaction. This value may subsequently be modified by a SET TRANSACTION RETRY LIMIT statement in an action diagram.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
RUNTIME-PARM1	input	This is DFHEIBLK automatically included by the Translator.
RUNTIME-PARM2	input	This is DFHCOMMAREA automatically included by the Translator.

Name	I/O	Description
RETRY-TIMES	input/ output	The maximum number of times the transaction execution is retried.
GLOBDATA	input	Global data, used internally.

Return Code

No explicit return code is set by the user exit.

Default Processing

If the Default Retry Limit Exit is not modified the TRANSACTION RETRY LIMIT will be initialized to 10 for all target environments. If the Default Retry Limit Exit is used, it must not return a value greater than that specified in the Ultimate Retry Limit Exit.

Customizing the Exit

The TRANSACTION RETRY LIMIT will be initialized to this value at the beginning of each new transaction. This value may subsequently be modified by a SET TRANSACTION RETRY LIMIT statement in an action diagram.

The TRANSACTION RETRY LIMIT is used to specify the maximum number of times a transaction is to be retried when one of the following events occurs:

- A RETRY TRANSACTION action diagram statement executes.
- A deadlock or timeout occurs trying to access a database, and no WHEN DATABASE DEADLOCK OR TIMEOUT statement was provided for that entity action statement.

In either of these cases, any uncommitted database updates will be rolled back, and an attempt will then be made to execute the application again. Once the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, no more retries can occur, and the application will fail with a runtime error if the last retry attempt was unsuccessful.

Modify the copied exit as needed. When you have completed your modifications, install the exit as described in Customizing and Installing z/OS User Exits.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exit

TIRBURTX

TIRBURTX—Ultimate Retry Limit Exit

Ultimate Retry Limit Exit for batch applications.

Source Code

LINKAGE SECTION.

```
01  ULTIMATE-RETRY-LIMIT      PIC S9(9)   COMP.
```

Purpose

The Ultimate Retry Limit Exit allows the user to specify a maximum value for the TRANSACTION RETRY LIMIT system attribute. This value may never be exceeded, either by a SET TRANSACTION RETRY LIMIT statement in an action diagram, or by the Default Retry Limit Exit.

For an explanation of when and how the TRANSACTION RETRY LIMIT system attribute is used see Default Retry Limit Exit in this chapter.

Once the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit Exit, no more retries can occur, and the application will fail with a runtime error if the last retry attempt was unsuccessful.

Default Processing

If the Ultimate Retry Limit Exit is not used, the maximum value of TRANSACTION RETRY LIMIT will be 99 for all target environments. The Ultimate Retry Limit Exit may be modified to return a value of zero to suppress all retry attempts for that environment.

Customizing the Exit

Copy TIRBURTX exit to one of your libraries or directories. Modify the copied exit as needed. When you have completed your modifications, install the exit as described in the Customizing and Installing z/OS User Exits.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

TIRRETCX—Batch Return Code Override Exit

TBD

Source Code

TBD

Purpose

The Batch Return Code Override Exit allows the user to override the CA Gen defined COBOL return code in batch job steps. It is called by the Batch Manager at the end of each job step, and by program TIRIOVFI.

The COBOL return code is used by CA Gen to implement transfer dialog flows in batch jobs. Due to the implications of the return code on job step execution, be sure you understand the behavior of JCL condition codes before modifying this exit.

The return code may be safely overridden if either:

- There is only one procedure step executed by the job, or
- All exit states in the job's procedure steps cause a transfer either to the same procedure step (self-referencing flow) or to the next procedure step (no job steps are bypassed).

Under these conditions, all steps in the job are executed in order and the condition code may be safely set to another value (e. g., zero). If this is not the case, you must restructure the batch procedure or modify the JCL to prevent improper attempted execution of the job steps. An attempt to execute a job step without the proper control data in the TIRIOVF file will result in an ABEND.

Note that return codes are not used in the TSO testing of batch applications. Overflowing the return code will have no effect on the TSO testing of batch procedures.

Arguments

TBD

Return Code

TBD

Default Processing

The default exit takes no action.

Customizing the Exit

Copy the default exit from the sample library to one of your own libraries. The member name is TIRRETCX. The COBOL return code is passed to the exit as a parameter along with the names of the procedure step just completed and the load module in which it is packaged. For transfers, the names of the destination procedure step and load module are also provided.

Modify the copied exit as needed. When you have completed your modifications, install the exit as described in the section on customizing user exits in this chapter.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

TBD

TIRTERBX—Batch Termination Exit

TBD

Source Code

TBD

Purpose

Batch runtime errors are handled by the Batch Manager that is a part of every CA Gen generated batch load module.

When an error occurs, the batch job is terminated. The Batch Manager executes a fail routine that backs out changes by performing the necessary rollbacks of the databases. The fail routine then calls a termination exit that determines what diagnostic (error) information is written to the error file.

Arguments

TBD

Return Code

TBD

Default Processing

If a batch runtime error occurs and the default batch termination exit is used, processing is as follows:

1. The Batch Manager performs all necessary rollbacks. This is done regardless of the termination exit used.
2. The Batch Manager fail routine calls the default termination exit. It returns to the fail routine without doing anything, which causes the default termination logic in the fail routine to be used.
3. The fail routine writes runtime error messages to an error message file with the DDNAME TIRERRF.

   An example of an error message file is shown in the following figure.
4. The batch job abends with a user code of 100 (U0100).

```
TIRM030E:   APPLICATION FAILED ** UPDATES HAVE BEEN BACKED OUT
TIRM031E:   FAILING PROCEDURE EXIT DATA FOLLOWS
TIRM032E:   LAST OR CURRENT ACTION BLOCK ID    =  507774696
TIRM033E:   LAST OR CURRENT ACTION BLOCK NAME  =  ABADDEMP
TIRM034E:   LAST OR CURRENT DATABASE STATEMENT =
TIRM035E:   CURRENT STATEMENT BEING PROCESSED = 10
TIRM037E:   ** A FATAL ERROR HAS BEEN ENCOUNTERED **
TIRM047E:   APPLICATION FAILED ** MUST ABEND
TIRM048E:   WILL ABEND VIA U0100
```

Customizing the Exit

Copy member TIRTERBX from the CA Gen CEHBSAMP library to a separate library. The status code returned from TIRTERBX is used to control abend processing as shown in the following table:

Status Code Returned to TIRTERBX	TIRFAIL Writes Messages to Error File	TIRFAIL Abends Job with U0100
0	Yes	Yes
1	No	Yes
2	No	No
(blank)	Yes	Yes

CA Gen batch jobs usually handle errors by first writing error messages to a file and then abending with a code of U0100. Upon entry to TIRTERBX, the TERM-STATUS-CODE is blank. By setting this field to 1, you can override the error message processing but TIRFAIL still abends the job with a code of U0100. You can override the message processing and control the abend of the job by setting the TERM-STATUS-CODE to 2.

Important! If you set TERM-STATUS-CODE to 1 or 2, CA Gen does not write error messages to a file. You should implement your own routine to write error messages to a file so that you can see the nature of any errors that occur. Be sure to include the use of DSNTIAR for display of DB2 messages. If you set TERM-STATUS-CODE to 2, you are responsible for abending the job.

One way of doing this is to code:

CALL 'TIRABEND' USING ABEND-CODE

where ABEND-CODE is defined as:

01 ABEND-CODE PIC S9(9) COMP VALUE +100

The termination exit source contains example code for each of these status codes, with the code for 1 and 2 commented out. The parameters passed between the fail routine and the termination exit are defined in copy member CBLTERM, which is also in the CEHBSAMP library.

When you have completed your modifications, install the exit as described earlier in the section on customizing user exits in this chapter.

Building on z/OS

For more information about how to install this exit, see MKUEXITS in Customizing and Installing z/OS User Exits.

Related User Exits

None.

Customizing and Installing z/OS User Exits

The easiest way to customize a user exit is to copy and modify the code supplied by CA. Always read the source code of the exit as well as the additional information included in comment lines. These lines are placed where you would need to enter code to customize the exit for a specific need or function.

The steps involved in customizing any user exit are as follows:

- Copy the default exit from the CA Gen CEHBSAMP library to one of your libraries.
- Add the desired new code to the copied exit. Do not change the 'Program ID' (entry point) or argument list passed to the exit.

Most user exits are called with RUNTIME-PARM1 and RUNTIME-PARM2 as the first two parameters. Under IMS, the RUNTIME-PARM parameters are mapped to the IO-PCB and ALT-IOPCB, respectively. If an exit is modified to use the IO-PCB and/or the ALT-IOPCB, remember to remove the corresponding RUNTIME-PARM from the LINKAGE SECTION and the PROCEDURE DIVISION USING statement.

Under CICS, the RUNTIME-PARM parameters are mapped to the CICS EXEC Interface Block (EIB) and the COMMAREA, respectively. If an exit is modified to use DFHEIBLK and/or DFHCOMMAREA remember to remove the corresponding RUNTIME-PARM from the LINKAGE SECTION and the PROCEDURE DIVISION USING statement. In addition, remove these parameters if the modified exit is processed by the CICS translator as the CICS translator automatically includes a reference to DFHEIBLK and DFHCOMMAREA.

Information about the presence and use of RUNTIME-PARM parameters is included in the CEHBSAMP member for each exit and is covered in this chapter if relevant.

The remaining parameters are unique to each exit and should not be modified.

Ensure the new runtime modules are placed in the proper data sets as follows:

- CICS – in the CICS DFHRPL concatenation and is New Copied into CICS.
- IMS – in the steplib concatenation.

As delivered, the Exits contain no SQL and no DBRMs. If a User Exit calls another routine that does database access, you need to add that routine's DBRM to the installation control file so the routine can be found. The best way to accomplish this is to modify the Installation CLIST, TICINSTX, so that it adds the DBRM to the DBRM list.

The following table identifies the z/OS JCL procedures used to compile and bind each of the z/OS Server Manager User Exits into their respective DLLs. These JCL procedures are located in the CA Gen CEHBSAMP library.

Each DLL may contain other user exits that are not directly in support of the processing of a z/OS Distributed Processing Server (DPS). Additionally, the list of z/OS user exits may also be used to support z/OS blockmode and batch operation.

JCL Procedure	DPS User Exit Name	Description
MKUEXITS		Incorporates customized user exits into the corresponding z/OS DLLs. The user exits names and corresponding z/OS DLL names can be found in the table in Changes to User Exits (see page 323).
MKCRUN	TIRXINFO	Incorporates customized user exits into the following z/OS DLLs: TIRCRUNC TIRCRUNI

Details of the preceding JCL procedures follow. Each one is described in a separate section.

MKUEXITS—Make COBOL Runtimes (User Exits DLLs)

MKUEXITS is a JCL Procedure that can be used to change the user exits used by the CA Gen Dynamic runtime DLLs listed in the table in Changes to User Exits (see page 323). These Dynamic runtime DLLs are used by z/OS blockmode applications and servers.

Source Code

The source code for this procedure is in CA Gen CEHBSAMP library, in member MKUEXITS.

Purpose

This procedure is used to modify the COBOL user exits used by the COBOL CA Gen Dynamic runtime DLLs.

Arguments

This procedure does not contain any arguments.

Return Code

Return code is not applicable.

Default Processing

The default processing of the JCL procedure is as follows:

- The CROBJLIB step creates a temporary dataset for the objlib.
- The *user exit* (ex. TIRALLOX) step invokes the COBOL compile proc to compile the modified user exit(s). Each user exit contained in a COBOL runtime DLL has a compile step.
- The COMPILE step compiles the user exit member(s) for inclusion in the runtime DLL.
- The linkedit step (that is, LKTERMA, LKMTQB, ….) links the user exit objects into a new user exit COBOL runtime DLL (that is, TIRTERMAZ, TIRMTQBZ, …).

Customizing the Exit

There can only be one user exit runtime DLL (that is, TIRTERAZ, TIRMTQBZ…) in the specific target region (CICS, IMS, TSO, BATCH). There are notes in the sample procedure with specific information about customizing the JCL. Various steps in the procedure can be modified.

- Not all user exits need to be modified and replaced. If a user exit compile step is removed, then the corresponding REPLACE and INCLUDE statements must be removed for the same user exit(s).
- Ensure that the data set pointed to the SYSLMOD statements for linkedit steps (that is, LKTERMA, LKMTQB, ….) are of type Library since the user exit runtime DLLs are a Program Object (format 3).

Building on z/OS

This is a JCL procedure that after modification can be submitted to produce the customized user exit runtime DLLS (that is, TIRTERAZ, TIRMTQBZ, ….) COBOL runtime DLLs containing user exits. A Condition Code of zero is expected for each step.

MKCRUN—Make C Runtimes - TIRCRUNC (CICS) and TIRCRUNI (IMS)

MKCRUN is a JCL Procedure that can be used to change the code page used by the CA Gen Dynamic runtime DLLs TIRCRUNC and TIRCRUNI. This Dynamic runtime is only used by z/OS servers.

Note: This procedure replaces procedure MKTIRE used by previous releases of CA Gen to build TIRENTC and TIRENTI. In addition, CA Gen DLLs TIRCRUNC and TIRCRUNI replace CA Gen runtimes TIRENTC and TIRENTI respectively.

Source Code

The source code for this procedure is in CA Gen CEHBSAMP library, in member MKCRUN.

Purpose

This procedure is used to modify the code page used by the CA Gen Dynamic runtime DLLs.

Arguments

This procedure does not contain any arguments.

Return Code

Return code is not applicable.

Default Processing

The default processing of the JCL procedure is as follows:

- The CA Gen step builds the list of translation code page pairs to be used by the CA Gen Runtimes. The first three code page pairs are the CA Gen default, the USA EBCDIC – ASCII, and the AMERICAN EBCDIC - MICROSOFT ASCII translation tables. These three code page pairs are always required so they must not be deleted, however the comments on the lines must be removed before the step is executed. A number of other code page pairs are listed, each of which is used for specific NLS translation. Update the list to keep only those code page pairs required, ensuring that any comments are removed.
- The TIRXINFO step assembles the user exit TIRXINFO. The source for user exit TIRXINFO can be found in CEHBSAMP.
- The GXTABLE step assembles the table produce in the GEN step.
- The LKCICS step links the subroutines modified in the previous steps into a new TIRCRUNC runtime DLL.
- The LKIMS step links the subroutines modified in the previous steps into a new TIRCRUNI runtime DLL.
- The LKCICSD step links the subroutines modified into a new TIRCRUNC debug runtime DLL.
- The LKIMSD step links the subroutines modified into a new TIRCRUNI debug runtime DLL.

Customizing the Exit

There can only be one TIRCRUNC DLL in a CICS region and one TIRCRUNI DLL in an IMS system. There are notes in the sample procedure with specific information about customizing the JCL. Various steps in the procedure can be modified:

- Code Page pairs can be added to the GEN Step. Ensure that the comments next to the code page pairs are removed before executing the procedure.
- The DEFCODEP and the DEFPADCH can be set to the required values in the TIRXINFO exit. More information about this exit is covered under TIRXINFO exit in this section.
- Ensure that the data set pointed to the SYSLMOD statements for steps LKCICS and LKIMS are of type Library since TIRCRUNC and TIRCRUNI are Program Objects (format 3).
- Optionally steps LKCICSD and LKIMSD can be executed to customize the debug versions of TIRCRUNC and TIRCRUNI respectively. Again, ensure the SYSLMOD statements points to a Library.

Building on z/OS

This is a JCL procedure that after modification can be submitted to produce the customized TIRCRUNC or TIRCRUNI runtime DLLs. A Condition Code of zero is expected for each step.

MKUECTCP—Make CICS TCP/IP Exits (TIRSLEXT and TIRSLTMX)

MKUECTCP is a JCL Procedure that can be used to change the user exits used by the CA Gen CICS TCP/IP Listener.

Source Code

The source code for this procedure is in CA Gen CEG8SAMP library, in member MKUECTCP.

Purpose

This procedure is used to modify the Assembler user exits used by the CICS TCP/IP Listener.

Arguments

This procedure does not contain any arguments.

Return Code

Return code is not applicable.

Default Processing

The default processing of the JCL procedure is as follows:

- The CROBJLIB step creates a temporary dataset for the objlib.
- The user exit (ex. TIRSLEXT) step invokes the Assembler proc to assemble the modified user exit(s). Each user exit has an assemble step.
- The LKSLEXT step links the user exit object into a new TIRSLEXT Assembler runtime exit for the CICS Listener.
- The LKSLTMX step links the user exit object(s) into a new TIRSLTMX Assembler runtime exit CICS Listener.

Customizing the Exit

There can only be one TIRSLEXT and TIRSLTMX exit in the CICS region. There are notes in the sample procedure with specific information about customizing the JCL. Various steps in the procedure can be modified.

- Not all user exits need to be modified and replaced. If a user exit compile step is removed, then the corresponding REPLACE and INCLUDE statements must be removed for the same user exit(s).
- Ensure that the data set pointed to the SYSLMOD statements for steps LKSLEXT and LKSLTMX are of type Library.

Building on z/OS

This is a JCL procedure that after modification can be submitted to produce the customized TIRSLEXT and TIRSLTMX CICS TCP/IP user exits. A Condition Code of zero is expected for each step.

MKUEITCP—Make IMS TCP/IP Exits (TIRxxTD, TIRxxTDC, and TIRxxTSC)

MKUEITCP is a JCL Procedure that can be used to change the user exits used by the CA Gen IMS TCP/IP exits. These exits are used by the CA Gen IMS connect exits TIRxxTCP, where xx represents the IMS Connect version.

Source Code

The source code for this procedure is in CA Gen CEG9SAMP library, in member MKUEITCP.

Purpose

This procedure is used to modify the Assembler user exits called by Gen's IMS TCP/IP exit provided for IMS Connect.

Arguments

This procedure does not contain any arguments.

Return Code

Return code is not applicable.

Default Processing

The default processing of the JCL procedure is as follows:

- The CROBJLIB step creates a temporary dataset for the objlib.
- The user exit (ex. TIR10TD) step invokes the Assembler proc to assemble the modified user exit(s). Each user exit has an assemble step.
- The LKG10TCP step links the user exit object(s) into a new TIR10TCP Assembler runtime exit for IMS 10.
- The LKG11TCP step links the user exit object(s) into a new TIR11TCP Assembler runtime exit for IMS 11.
- The LKG12TCP step links the user exit object(s) into a new TIR12TCP Assembler runtime exit for IMS 12.
- The LKG13TCP step links the user exit object(s) into a new TIR13TCP Assembler runtime exit for IMS 13.

Customizing the Exit

There can only be one CAGRITCP in IMS Connect so only the link step that corresponds to the version of IMS Connect being run in your shop should be executed. There are notes in the sample procedure with specific information about customizing the JCL. Various steps in the procedure can be modified.

- Not all user exits need to be modified and replaced. If a user exit compile step is removed, then the corresponding REPLACE and INCLUDE statements must be removed for the same user exit(s).
- Ensure that the data set pointed to the SYSLMOD statements for steps TIR10TCP, TIR11TCP, TIR12TCP, and TIR13TCP are of type Library.
- Rename the customized TIRxxTCP exit to CAGRITCP before deploying in IMS Connect.

Building on z/OS

This is a JCL procedure that after modification can be submitted to produce the customized TIR10TCP, TIR11TCP, TIR12TCP, and TIR13TCP IMS TCP/IP exit containing user exits. A Condition Code of zero is expected for each step. Rename the customized TIRxxTCP exit to CAGRITCP before deploying in IMS Connect.

Chapter 5: NonStop User Exits

There are several sets of user exits to support the variety of C applications that run on the NonStop platform. Scripts are provided to assist in building the runtime user exits for each runtime environment listed in the following sections.

The following table lists the sets of C user exits available on the NonStop platform.

User Exit Set	Provided As
Blockmode Runtime	UEXITCO
Server Runtime	UEXITCO, DPSUECO

NonStop Blockmode User Exits

The following table summarizes the functions available through the user exits for generated blockmode applications:

Name	Description
TIRDLCT	Dialect User Exit
TIRDRTL	Default Retry Limit User Exit
TIRHELP	Help Interface User Exit
TIRMTQB	Message Table User Exit
TIRSECR	Security Interface User Exit
TIRSYSID	System ID User Exit
TIRTERMA	User Termination User Exit
TIRUPDB	MBCS Uppercase Translation User Exit
TIRUPPR	Uppercase Translation User Exit
TIRURTL	Ultimate Retry Limit User Exit
TIRUSRID	User ID User Exit
TIRYYX	Date User Exit

All blockmode runtime user exits are provided in both source and object format (in the object library UEXITCO), and are rebuilt using the TACL macro MKEXITS which can be found in the subvolume where the IT has been installed. Details on the use of the MKEXITS macro can be found in the *NonStop Implementation Toolset User Guide*.

The UEXITCO user exit object library is the same one that is used with server applications.

Details for the preceding user exits follow in a separate section for each.

TIRDLCT—Dialect Exit

```
void TIRDLCT (
    char *rp1,
    char *rp2,
    struct dialect_cmcb *tirdlct_cmcb)
```

Source Code

TIRDLCT

Purpose

TIRDLCT supplies the current user's dialect to the application and is useful only for multilingual applications. For multilingual support, the user is responsible for modifying this module to return the appropriate dialect. The dialect returned should be defined using the Design selection on the CA Gen action bar. If it is not, the application's default dialect is used.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*tirdlct_cmcb	Input/Output	A pointer to a structure containing the following items:
tirdlct_userid	Input	An 8-byte character array containing the current user id as provided by TIRUSRID.

Name	I/O	Description
tirdlct_terminal_id	Input	An 8-byte character array containing the current terminal id.
tirdlct_system_id	Input	An 8-byte character array containing the current system id as provided by TIRSYSID.
tirdlct_return_dialect	Input	An 8-byte character array containing the returned dialect.

Return Code

None

Default Behavior

TIRDLCT returns a dialect value of DEFAULT.

Building on NonStop

The Dialect User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library

Action	Description
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

TIRUSRID, TIRSYSID

TIRDRTL—Default Retry Limit Exits

```
int tirdrtl (
    char retry_flag)
```

Source Code

TIRDRTL.C

Purpose

TIRDRTL lets you override the CA Gen-defined default value for the TRANSACTION RETRY LIMIT system attribute. TRANSACTION RETRY LIMIT will be initialized to this value at the beginning of each new transaction. This value can subsequently be modified by a SET TRANSACTION RETRY LIMIT statement in an action diagram.

TRANSACTION RETRY LIMIT is used to specify the maximum number of times to retry a transaction when one of the following events occurs:

- A RETRY TRANSACTION action diagram statement executes.
- A deadlock or timeout occurs trying to access a database, and there is no WHEN DATABASE DEADLOCK OR TIMEOUT statement for that entity action statement.

In these cases, uncommitted database updates are rolled back, and an attempt is made to execute the application again. After the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit User Exit (see TIRURTL), no more retries can occur, and the application fails with a runtime error.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
retry_flag	Input	Flag to indicate whether or not to set a retry limit.

Return Code

Integer containing the retry limit.

Default Behavior

If the Default Retry Limit User Exit is not used, TRANSACTION RETRY LIMIT will be initialized to 10 for all target environments. If the Default Retry Limit User Exit is used, it must not return a value greater than that specified in the Ultimate Retry Limit User Exit.

Building on NonStop

The Default Retry User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library

Action	Description
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

TIRURTL

TIRHELP—Help Interface Exit

```
void TIRHELP (
    char *rp1,
    char *rp2,
    struct tirhelp *in_tirhelp_cmcb,
    char *in_tirhelp_return_message,
    char *in_environment_list,
    char *in_application_list,
    struct scmgr *in_scmgr_cmcb)
```

Source Code

TIRHELP.C

Purpose

TIRHELP is called when a HELP or PROMPT command is entered. From TIRHELP, a help system can be invoked to provide application help information.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.

Name	I/O	Description
*in_tirhelp_cmcb	Input/Output	A pointer to a structure containing the following items:
tirhelp_request_code	Input/Output	A 2-byte character array containing the type of help requested.
tirhelp_return_code	Output	A 2-byte character array containing the return code.
tirhelp_mapname	Input	An 8-byte character array containing the mapname.
tirhelp_data_filler	Unused	An 8-byte character array used as a structure filler.
tirhelp_trancode	Input	An 8-byte character array containing the trancode.
tirhelp_userid	Input	An 8-byte character array containing the user id.
tirhelp_terminal_id	Input	An 8-byte character array containing the terminal id.
tirhelp_printer_id	Input	An 8-byte character array containing the printer id.
tirhelp_dialect	Input	An 8-byte character array containing the dialect.
tirhelp_message_table	Input	An 8-byte character array containing the message table. This value is passed to TIRMTQB.
tirhelp_filler	Unused	A 16-byte character array used as a structure filler.
tirhelp_last_command	Input	An 80-byte character array containing the last command.
tirhelp_last_message	Input	An 80-byte character array containing the last message.
tirhelp_screen_helpid	Output	A 44-byte character array containing the help identifier for the screen.
tirhelp_field_helpid	Output	A 44-byte character array containing the help identifier for the field.
tirhelp_field_token1	Input	A 3-byte character array containing a field token.
tirhelp_field_token2	Input	A 3-byte character array containing a second field token
tirhelp_field_len	Input	A 3-byte character array containing the field length.
tirhelp_field_value	Input	A 256-byte character array containing the value of the field.
tirhelp_field_protect	Input	A single character containing a field protection flag.
tirhelp_field_intens	Input	A single character containing a field intensity flag.

Name	I/O	Description
in_tirhelp_return_message	Output	An 80-byte character array representing the returned help message. By default, this message is returned from a call to TIRMTQB.
in_environment_list	Input	A pointer to an environment control block. Reserved for runtime internal use only.
in_application_list	Input	A pointer to an application control block. Reserved for runtime internal use only.
in_scmgr_cmcb	Input/Output	A pointer to a screen management control block.

Return Code

None

Default Behavior

The TIRHELP routine will return a message indicating no help is available.

Building on NonStop

The Help Interface User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.

Action	Description
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

TIRMTQB

TIRMTQB—Message Table Exit

```
void TIRMTQB(char *rp1,
    char *rp2,
    char *msg_tbl_name,
    short *msgnum,
    struct PARMMSG *prm);
```

Source Code

TIRMTQB.C

Purpose

This message table exit is called by the runtime when a system-level message is to be displayed. The user can customize the wording of the messages within this exit. Additional tables can also be defined to support other dialects.

The default table includes an entry for each CA Gen runtime error message. Each entry includes the following information:

- Message Number—The message number is permanently assigned by CA Gen. Each message has a unique number.
- Message Text—The message text is the actual words that appear on the application screen when an error occurs. The message text, and any variable values that can be appended, is truncated if it exceeds the length of the error message line defined for the application screen. The error message line is a maximum of 80 characters of which 12 are reserved for the message number.

If the message number is not in the table, TIRMTQB returns a default message.

Runtime Error Table

The Runtime Error Message Table includes an entry for each runtime error message. Each table entry includes the following information:

- Message type— a message number is not found in the table, or when you return to a transaction screen after a fatal error or a Dialog Manager error occurs. Valid message types are shown in the following list:
- Default message— a message number is not found in the table, or when you return to a transaction screen after a fatal error or a Dialog Manager error occurs.
- Dialog Manager error—Occurs when the Dialog Manager is unable to communicate with the system. This is a fatal error beyond the control of CA Gen. An error in the load module packaging or in the configuration specifications causes a Dialog Manager error. Error handling is the same as for a fatal error.
- Fatal error— a CA Gen application abnormal program ending. If a condition occurs at runtime that the generated code cannot handle, the system issues a fatal error. An error message screen displays the appropriate error messages.
- Function error—Occurs if a CA Gen-supported function receives invalid input or produces invalid output. CA Gen-supplied functions manipulate characters, numbers, dates, and times.
- Screen edit error—A non-fatal error that occurs when an input or output value for a field does not match the expected value, the range, type, or format defined for the field during model development. This type of message is displayed on your transaction screen. You can correct the error and continue with the transaction.
- Unformatted input error—Occurs when the unformatted input contains invalid parameters, delimiters, or both. Unformatted input is a list of parameters associated with a clear screen transaction code.
- Message number—Each message has a unique number that is permanently assigned by CA Gen.
- Message text—The message text consists of the actual words that appear on the application screen when an error occurs. Because of the length of the message identifier, the message text is limited to 68 characters for an 80-character screen. The message text and appended variables are truncated if they exceed the length of the error message line defined for the application screen.
- Suffix—(If applicable) The suffix contains variable values, such as return codes, permitted values, or the values in error.

Runtime Error Handling

Runtime errors are handled by the Dialog Manager. Runtime errors are non-fatal, such as screen edit, or fatal errors.

If a non-fatal error such as invalid user input occurs, the Dialog Manager displays an error message on the transaction screen. You can correct the error and continue processing the transaction.

If an application fails because of a fatal error, transaction processing terminates, and the error processing is as follows:

- The Dialog Manager performs all necessary rollbacks of the databases.
- CA Gen displays an error message screen that lists the appropriate runtime error messages.
- Pressing Enter from the error message screen causes CA Gen applications to terminate execution.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*msg_tbl_name	Input	A character string containing the name of the table to be used for extraction of the message text. Currently one table named DEFAULT is supported by the CA Gen runtime.
*msgnum	Input	A short value containing the message number corresponding to the text to be fetched.
*prm	Input/ Output	A pointer to a PARMMSG structure to contain the returned message text information. This structure, defined in tirmtq.h, has the following definition:
PARMLEN		A short value containing the total length of PARMNO + PARMTXT.
PARMNO	Output	An 11-byte character array containing the message number formatted in a standard style.
PARMTXT	Output	A string containing the text corresponding to the error message number. The string can be up to 245 bytes, including the terminating NULL.

Name	I/O	Description
filler	Unused	Two unused filler characters

Return Code

None

Default Behavior

The table in the default exit is used to retrieve runtime error message text.

Building on NonStop

The Message Table User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings

Action	Description
Q	To exit MKEXITS

Related User Exits

None

TIRSECR—Security Check Interface Exit

```
void TIRSECR(char * rp1,
    char * rp2,
    struct security_cmcb * in_tirsecr_cmcb);
```

Source Code

TIRSECR.C

Purpose

The Dialog Manager calls the Security Check Interface Exit when a transaction is started and before execution of a dialog flow. This allows transaction-level security checking to be implemented. The following data is provided by the dialog manager of each load module for use in checking security authorization:

- System ID (as provided by the System ID Exit, TIRSYSID)
- User ID (as provided by the User ID Exit, TIRUSRID)
- Trancode
- Terminal ID
- Load module name
- Procedure step name

If the user defined security check passes, TIRSECR should move a value of spaces to the return code. If the security check fails, a non-blank value should be moved to the return code with a message describing the violation inserted into the tirsecr_failure_msg buffer. The current dialect in effect on the client is passed in using tirsecr_dialect.

When the dialog manager receives control, it proceeds with the transaction if the return code is spaces, or issues an error if it is not.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*in_tirsecr_cmcb	Input/Output	A pointer to a structure containing the following items:
tirsecr_userid	Input	An 8-byte character array containing the security user ID as provided by the TIRUSRID user exit
tirsecr_trancode	Input	An 8-byte character array containing the current transaction code.
tirsecr_terminal_id	Input	An 8-byte character array containing the current terminal ID.
tirsecr_system_id	Input	An 8-byte character array containing the current system ID as returned by the TIRSYSID user exit.
tirsecr_load_module	Input	An 8-byte character array containing the name of the executing load module calling this exit.
tirsecr_pstep_name	Input	A 32-byte character array containing the name of procedure step being executed.
tirsecr_dialect	Input	A 32-byte character array containing the dialect in effect on the client.
tirsecr_return_code	Output	A 2-character array representing the success or failure of this exit processing. TIRSECR_ALL_OK defined as two spaces implies success, any other value implies failure. If none spaces are return, tirfail will be passed the tirsecr_failure_msg message.
tirsecr_failure_msg	Output	An 80-byte character array used in conjunction with a failing return code in tirsecr_return_code. This exit can insert an error message into this array that will be passed by the Dialog manager to the tirfail user exit.

Return Code

None directly. For more information, see tirsecr_return_code structure member.

Default Behavior

The default exit will return a status code of spaces, indicating no security violation was detected.

Building on NonStop

The Security Check User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

TIRUSRID, TIRSYSID

TIRSYSID—System ID Exit

```
void TIRSYSID ( char *rp1;
    char *rp2;
    char *system_id);
```

Source Code

TIRSYSID.C

Purpose

TIRSYSID supplies the system ID to the application.

The purpose of TIRSYSID is to implement application logic that lets you implement one model on multiple platforms, and perform processing appropriate for the platform. The system ID is also one of the parameters passed to the Security Interface Exit (TIRSECR).

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*system_id	Output	An 8-byte character array representing the system identifier where the server application is executing.

Return Code

None

Default Behavior

By default, TIRSYSID calls the runtime routine DEFSYSID. This routine returns a default system ID, the value of which depends on the platform on which the application is executing.

Under UNIX/Linux if the environment variable IEF_SYSID is set the first 8 characters of this variable are used. Otherwise, “UNIX“ is returned.

Building on NonStop

The System ID User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

TIRSECR

TIRTERMA—User Termination Exit

```
void TERTERMA (
    char *rp1,
    char *rp2,
    struct term_pb *pb)
```

Source Code

TIRTERMA.C

Purpose

TIRTERMA is called when an application fails. Modification of TIRTERMA lets the user customize the handling of runtime errors.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*pb	Input/ Output	A pointer to a PARMMSG structure to contain the termination information. This structure is defined in tirterma.h.

Return Code

None

Default Behavior

The default processing for TIRTERMA returns a status code of spaces, indicating to use standard error handling.

Building on NonStop

The Termination User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

None

TIRUPDB—MBCS Uppercase Translation Exit

```
void TIRUPDB (
    char *rp1,
    char *rp2,
    char *tbl_name,
    long *len,
    char *xlate_data)
```

Source Code

TIRUPDB.C

Purpose

TIRUPDB is called to uppercase multi-byte text. The user can modify the mechanism used to uppercase multi-byte text with this user exit.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*tbl_name	Input	A pointer to a translation table name.
*len	Input/Output	Length of text to convert to uppercase.
*xlate_data	Input/Output	A pointer to the text to be uppercased.

Return Code

None

Default Behavior

The default translation uses MBCS functions to perform uppercase translation based upon the active system code page. However, the system designer, programmer, may add code to recognize dialects and perform any lower to upper functionality desired. In that case, insure that the default behavior still uses the MBCS libraries.

Building on NonStop

All the user exits in this section are built as part of the object library UEXITCO unless specified otherwise. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

TIRUPPR

TIRUPPR—Uppercase Translation Exit

```
void TIRUPPR (
    char *rp1,
    char *rp2,
    char *tbl_name,
    long *len,
    char *xlate_data)
```

Source Code

TIRUPPR.C

Purpose

TIRUPPR is called to uppercase multi-byte text. The user can modify the mechanism used to uppercase multi-byte text with this user exit.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*tbl_name	Input	A pointer to a translation table name.
*len	Input/Output	Length of text to convert to uppercase.
*xlate_data	Input/Output	A pointer to the text to be uppercased.

Return Code

None

Default Behavior

The default translation uses MBCS functions to perform uppercase translation based upon the active system code page. However, the system designer, programmer, may add code to recognize dialects and perform any lower to upper functionality desired. In that case, insure that the default behavior still uses the MBCS libraries.

Building on NonStop

The Uppercase Translation User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

TIRUPDB

TIRURTL—Ultimate Retry Limit Exit

```
long tirurtl ()
```

Source Code

TIRURTL.C

Purpose

TIRURTL lets you specify a maximum value for the TRANSACTION RETRY LIMIT system attribute. This value can never be exceeded by a SET TRANSACTION RETRY LIMIT statement in an action diagram, or by the Default Retry Limit User Exit.

After the number of retries, as indicated by the TRANSACTION RETRY COUNT system attribute, reaches either TRANSACTION RETRY LIMIT or the value specified by the Ultimate Retry Limit User Exit, no more retries can occur, and the application fails with a runtime error.

Arguments

None

Return Code

Long containing the retry limit.

Default Behavior

If the Ultimate Retry Limit User Exit is not used, the maximum value of TRANSACTION RETRY LIMIT will be 99 for all target environments. The Ultimate Retry Limit User Exit can be modified to return a value of zero to suppress all retry attempts.

Building on NonStop

The Ultimate Retry User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS

4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

TIRDRTL

TIRUSRID—User ID Exit

```
void TIRUSRID ( char *rp1;
    char *rp2;
    char *filler_parm;
    char *user_id);
```

Source Code

TIRUSRID.C

Purpose

TIRUSRID is used to supply the user's ID to the application. The user ID is one of the parameters passed to the Security Interface Exit (TIRSECR).

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*filler_parm	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*user_id	Output	A pointer to an 8-byte character array into which the user ID can be returned.

Return Code

None

Default Behavior

The default action taken by this module is to call runtime routine DEFUSRID which returns a default user ID, the value of which depends on the platform on which the system is executing.

Building on NonStop

The User ID User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.

Action	Description
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

TIRSECR

TIRYYX—Date Exit

```
void TIRYYX (
    struct tiryyx_param_block *pb)
```

Source Code

TIRYYX.C

Purpose

TIRYYX is used to process two-digit or yy-style date input and to set the century part using any fixed-window, sliding-window, or other algorithm of choice, when using CA Gen in the standard map generation mode.

Internally, CA Gen handles four digit year dates correctly assuming the user application uses the yyyy edit pattern throughout. If the user interface is designed to accept a two-digit date entry, and defaulting to the current century is not acceptable, use this exit to implement logic to get the required behavior for defaulting the century part of the date.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*pb	Input/Output	A pointer to a tiryyx structure containing the following items:
return_code	Output	A 4-byte character array containing the current year
current_year	Input	A 4-byte character array containing the current year.
edit_year	Input/Output	A 4-byte character array containing the edit year.

Return Code

None

Default Behavior

The default user exit behavior does not perform any processing and returns.

Building on NonStop

The Date User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.

Action	Description
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

None

NonStop Server User Exits

The following table summarizes the functions available through the user exits for generated server applications:

Name	Description
SRVRERROR	Server to Server Error User Exit (Server Only)
TIRDCRYP	Decrypt User Exit (Server Only)
TIRDLCT	Dialect User Exit
TIRDRTL	Default Retry Limit User Exit
TIRELOG	Server Error Logging User Exit (Server Only)
TIRHELP	Help Interface User Exit
TIRMTQB	Message Table User Exit
TIRNCRYP	Encrypt User Exit (Server Only)
TIRSECR	Security Interface User Exit
TIRSECV	Server Security Validation User Exit (Server Only)
TIRSYSID	System ID User Exit
TIRTERMA	User Termination User Exit
TIRUPDB	MBCS Uppercase Translation User Exit
TIRUPPR	Uppercase Translation User Exit
TIRURTL	Ultimate Retry Limit User Exit

Name	Description
TIRUSRID	User ID User Exit
TIRXINFO	Locale Information User Exit (Server Only)
TIRXLAT	National Language Translation User Exit (Server Only)
TIRYYX	Date User Exit
USEREXIT	Distributed Processing Flow Data Access User Exit (Server Only)

All server runtime user exits are provided in both source and object format (in the object library UEXITCO and DPSUECO), and are rebuilt using the TACL macro MKEXITS which can be found in the subvolume where the IT has been installed. Details on the use of the MKEXITS macro can be found in the *NonStop Implementation Toolset User Guide*.

The UEXITCO user exit object library is the same one that is used with blockmode applications.

Details for the preceding user exits follow in a separate section for each.

TIRDCRYP—Server Decryption Exit

```
void TIRDCRYP( unsigned char * rp1,
    unsigned char * rp2,
    TIRDCRYP_cmcb * pTIRDCRYP_cmcb);
```

Source Code

tirdcryp.c

Purpose

TIRDCRYP is called by the Server Manager after it detects that the client has sent an encrypted cooperative buffer. The Server Manager constructs a work buffer containing the concatenated View Data and Client Security sections. The user is responsible for decrypting the area pointed to by pDataBuffer for IBufferSize bytes.

The inputs pDataBuffer and IDecryptMaxSize as well as the outputs IBufferSize return_code and failure_msg are fields within a structure pointed to by the pTIRDCRYP_cmcb parameter.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only
*pTIRELOG_cmcb	Input/Output	A pointer to a TIRELOG_CMCB structure containing the following items:
lDecryptMaxSize	Input	A long field that contains the maximum available buffer space (in bytes) that the decrypted data can occupy.
lBufferSize	Input/Output	On input, lBufferSize is the current buffer space (in bytes) of the encrypted data. On output, lBuffferSize should be updated by this exit to contain the length of the decrypted data. The length of the decrypted result cannot exceed lDecryptMaxSize.
*pDataBuffer	Input/Output	On input, a pointer to the starting location of the encrypted View Data and Client Security sections within the CFB work buffer. On output, this exit should ensure this same data area contains the unencrypted versions of the input data. The length of this decrypted result cannot exceed lDecryptMaxSize.
return_code	Output	A two-character array returning the results of the decryption attempt. The following values are supported: DECRYPTION_USED—defined as " " DECRYPTION_SIZE_EXCEEDED_MAX—defined as "01" DECRYPTION_NOT_USED—defined as "02" DECRYPTION_APPLICATION_ERROR—defined as "03"
*failureMsg	Output	The pointer to an 80-character array, to be populated by the exit that can receive a null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of DECRYPTION_APPLICATION_ERROR.

Return Code

None directly, see the preceding return_code structure member.

Default Behavior

Decryption of the data buffer is not attempted.

Building on NonStop

The Server Decryption User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

The following are related user exits:

- TIRNCRYP
- WRSECENCRYPT

TIRELOG—Server Error Logging and Error Token Creation Exit

```
void TIRELOG(char * rp1,
    char * rp2,
    TIRELOG_CMCB * pTIRELOG_cmcb));
```

Source Code

tirelog.c

Purpose

This exit serves two purposes:

- Error logging at the server
- Creation of an error token for transmitting to the client

This exit is called by the server to handle server errors that are encountered during the execution of a distributed processing server that cannot be handled by the runtime or generated code, and normally result in the termination of the application. For example, prior to the execution of a server procedure step, the server extracts view data from the client message and places it in the target procedure step's view. If this extraction fails because of a mismatch between the client definition and the server definition an error response message is created and returned to the client.

The default implementation of this exit returns to the caller without logging the error. It is up to the developer of this user exit to determine what information should be logged and how it should be logged. Some users can choose to log only certain errors; others can choose to log all errors. On some systems, the log can be implemented as a file. To log a server error, simply format the information you wish to log and write it to a file. On other systems, the log can be implemented using system-specific features such as a CICS temporary storage queue (TSQ) as found on z/OS.

To create an error token, move text data to the area pointed to by the elog_error_token member of the TIRELOG_CMCB structure passed into this exit. The error token area is 4097 bytes and must be null-terminated. The error token, which goes through codepage translation when it is transmitted to the client, can be used on the client to customize how the error is handled.

For example, you can modify this exit to return an error token of "RETRY" whenever a certain database contention error occurs. This error token is passed to the client error-handling exit (WRSRVRERROR or WRASYNCSRVRERROR), which makes the final decision on how to handle the error. You can modify the client error-handling exit to reinvoke the flow or USE whenever the error token is "RETRY." This server error-logging exit is called after the error response message is created but before it is transmitted to the client.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only
*pTIRELOG_cmcb	Input/Output	A pointer to a TIRELOG_CMCB structure containing the following items:
elog_fail_type	Input	A character designating the type of failure detected defined as:(lable - defined value) EPROFD - 'P' profile error EPROFI - 'I' profile error EEXEC - 'E' execution error ESERVER - 'D' server manager error EUSER - 'U' user requested abend
void *elog_sqlca	Input	A pointer to a saved sql data area
*elog_globdata	Input	A pointer to the server's globdata area
elog_number_of_lines	Input	An integer containing the number of text lines contained within the elog_error_text buffer
elog_error_text	Input	A pointer to a buffer of screen formatted text. This data, formatted by the server runtime, contains up to 24 lines of 80 characters each.
*elog_error_token	Input/Output	A character pointer to an error token area that can contain up to 4097 bytes, this includes the required null terminator. This exit is responsible for populating this data area if needed.

Return Code

None directly, see the preceding pTIRELOG_cmcb structure.

Default Behavior

The default action is to return without logging the error.

Building on NonStop

The Server Error Logging User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

The following are related user exits:

- WRSRVRERROR
- WRASYNCSRVRERROR

TIRNCRYP—Server Encryption Exit

```
void TIRNCRYP( unsigned char * rp1,
    unsigned char * rp2,
    TIRNCRYP_cmcb * pTIRNCRYP_cmcb);
```

Source Code

tirncryp.c

Purpose

After a server procedure step executes, the server manager can call TIRNCRYP to encrypt the server response to the client. The server manager makes a copy of the unencrypted cooperative buffer pending transmission back to the client. The inputs pDataBuffer, IBufferSize, IEncryptMaxSize trancode and client_userid as well as the outputs return_code, and failure_msg are fields with a structure pointed to by pTIRNCRYP_cmcb. The user is responsible for encrypting the data area pointed to by the pDataBuffer member of the TIRNCRYP_cmcb structure.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*pTIRNCRYP_cmcb	Input/Output	A pointer to a structure containing the following items:

Name	I/O	Description
pDataBuffer	Input/Output	On input, a pointer to the starting location of the View Data and Client Security sections within the CFB work buffer. On output this same data area should be populate by this exit with the encrypted versions of the input data. The length of the encrypted result cannot exceed lEncryptMaxSize.
lBufferSize	Input/Output	On input, lBufferSize is the current buffer space (in bytes) of the unencrypted data. On output, lBuffferSize should be updated by this exit to contain the length of the encrypted data. The length of the encrypted result cannot exceed lEncryptMaxSize.
lEncryptMaxSize	Input	A long field that contains the maximum available buffer space (in bytes) that the encrypted data can occupy.
trancode	Input	Transaction code currently being processed. . This value can be used in conjunction with client userid and NextLocation to determine if encryption is desired.
client_userid	Input	Client user ID. This value can be used in conjunction with trancode and NextLocation to determine if encryption is desired.
pNextLocation	Input	Next Location value as set by the server application using CA Gen action diagram statements. This value can be used in conjunction with trancode and client userid to determine if encryption is desired.
return_code	Output	A two-character array returning the results of the decryption attempt. The following values are supported: ENCRYPTION_USED—defined as " " ENCRYPTION_SIZE_EXCEEDED_MAX—defined as "01" ENCRYPTION_NOT_USED—defined as "02" EnCRYPTION_APPLICATION_ERROR—defined as "03"
*failureMsg	Output	The pointer to an 80-character array, to be populated by the exit that can receive a null terminated error message string. The string pointed to by the failureMsg pointer will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of ENCRYPTION_APPLICATION_ERROR.

Return Code

None directly, see the preceding return_code structure member.

Default Behavior

The default logic of this user exit is to return ENCRYPTION_NOT_USED.

Building on NonStop

The Server Encryption User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

The following are related user exits:

- TIRDCRYP
- WRSECDECRYPT

TIRSECV—Security Validation Exit

```
void TIRSECV(char *rp1,
    char *rp2,
    unsigned char Enhanced_Security_Flag,
    PTIRSECV_cmcb pTIRSECV_cmcb);
```

Source Code

tirsecv.c

Purpose

This security exit is called for every cooperative flow, regardless of the security type used. To facilitate security validation a flag indicating whether the security data is for a standard or enhanced buffer has been added. This exit is intended to provide the opportunity to validate enhanced security data while at the same time not impacting those using standard security.

To this effect, the default code provided handles two possible conditions:

- For buffers containing standard security the client userid, client password, and security token fields are expected to be blank. The default behavior is for the exit to return SECURITY_USED, thus indicating that the request is authorized. The exit must be modified to return SECURITY_APPLICATION_ERROR if the intent is that all buffers contain enhanced security data.
- For buffers containing enhanced security the client userid, client password, and security token fields can or cannot contain data. The default behavior is for the exit to return SECURITY_NOT_USED, this indicating that no validation processing was attempted. The exit must be modified to validate the security data and set the relevant return code (return SECURITY_USED for an authorized user and SECURITY_APPLICATION_ERROR for a non authorized user). When returning SECURITY_APPLICATION_ERROR, this exit can provide an optional failure message, using the failure_msgbuffer contained within the TIRSECV_cmcb structure that will be presented to the client.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*rp1	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
*rp2	Input	A pointer to a parameter control block. Reserved for runtime internal use only.
Enhanced_Security_Flag	Input	A single character denoting if the CFB has been created to support enhanced security. A value of Y denotes enhanced security,
*pTIRSECV_cmcb	Input	A pointer to a structure containing the following values:
client_userid	Input	A 64-byte character array containing a user ID if the CFB uses enhanced security. For a CFB containing standard security this parameter is expected to be blank.
client_password	Input	A 64-byte character array containing a password if the CFB uses enhanced security. For a CFB containing standard security this parameter is expected to be blank.
lSecurityTokenLen	Input	A long value representing the length of the pSecurityToken, if any.
pSecurityToken	Input	A pointer to a security token if the CFB uses enhanced security. For a CFB containing standard security this parameter is expected to be blank.
trancode	Input	An 8-byte character array containing the transaction code
return_code	Input/Output	A 2-byte character array containing a value denoting success for failure of this exit. Valid values are: SECURITY_USED - defined as " " SECURITY_NOT_USED—defined as "02" SECURITY_APPLICATION_ERROR—defined as "03"

Name	I/O	Description
failure_msg	Input/Output	The pointer to an 80-character array that can be populated by this exit with a null terminated error message string. The string pointed to by this parameter will be incorporated into an error message that is returned back to the client. Used in conjunction with a return code of SECURITY_APPLICATION_ERROR.

Return Code

None directly, see the preceding return_code structure member.

Default Behavior

The default logic of this user exit is to return SECURITY_NOT_USED, which is considered an error if this user exit is actually called since the Server Manager requested Client Security validation.

Building on NonStop

The Server Security Validation User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.

Action	Description
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

WRSECTOKEN

TIRXINFO—Locale Information Exit

```
void TIRXINFO (char *osId,
    char *codePage,
    long *padChar);
```

Source Code

tirxlat.c

Purpose

This exit provides information about the codepage environment of the executing server process. An osId, codepage ID, and default padding character are returned. The runtime uses the osId and codePage returned as parameters passed into the TIRXLAT user exit.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*osId	Output	A pointer to character buffer to contain an OS ID (9 bytes, 8 characters plus NULL terminator). This value will be passed to TIRXLAT as the outOS parameter for inbound transactions and as the inOS parameter for outbound transactions. The current default value is MBCS. This should not be confused with an identifier of the underlying operating system on which the server is executing.

Name	I/O	Description
*codePage	Output	A pointer to character buffer to contain a codepage ID (9 bytes, 8 characters plus NULL terminator). This value will be passed to TIRXLAT as the outCodePage parameter for inbound transactions and as the inCodePage parameter for outbound transactions. The default value, as returned from this exit, is hard coded into the generated server manager at code generation time. This value will depend upon the platform used to generate the server manager. If the server manager is generated on a Windows platform the value will be 1252, if generated on a UNIX platform using CSE its value will be 819.
*padChar	Output	A pointer to a long value, not currently used for Windows or UNIX servers.

Return Code

None

Default Behavior

The string returned for osId is currently hard coded to a value of MBCS. The value for CodePage is obtained from the server manager. The CodePage number is created during the server manager code generation process. The padChar value is currently unused.

Building on NonStop

The Locale Information User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.

Action	Description
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

TIRXLAT

TIRXLAT—National Language Translation Exit

```
void TIRXLAT (char *inBuf;
    long *inLen;
    char *inCodePage;
    char *inOS;
    char *outBuf;
    long *outLen;
    char *outCodePage;
    char *outOS;
    long *outPadChar;
    char *workArea;
    long *outCharCnt;
    long *outByteCnt);
```

Source Code

tirxlat.c

Purpose

TIRXLAT allows the conversion of textual data based on from/to codepage and operating system information. View data that is passed between the client and server is translated from the client's code page to the server's code page, and vice versa. TIRXLAT uses the client's code page value, which is passed from the client to the server, and the host's code page value to locate a translation table.

This exit is used to translating both the data received from that client and the data to be sent to the client.

When translating data received from the client, the in* parameters correspond to the client data, the out* parameters correspond to the data presented to the server.

When translating data to be sent to the client the in* parameters correspond to the server data to be sent, the out* parameters correspond to the data presented to the client.

If a suitable translation table is not found, the data will be passed back without translation. The user can replace a translation table to customize their environment.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*inBuf	Input	A character pointer to the input buffer to translate
*inLen	Input	A pointer to a long value which is the length inBuf
*inCodePage	Input	A character pointer to the codepage ID of inBuf (8 bytes + 1 NULL).
*inOS	Input	A character pointer to the OS ID of inBuf (8 bytes + 1 NULL).
*outBuf	Input/Output	A character pointer to the buffer in which to place the translated text
*outLen	Input	A pointer to a long value that is the length of the data pointed to by outBuf.
*outCodePage	Input	A character pointer to the codepage ID corresponding to the output buffer, outBuf.
*outOS	Input	A character pointer to the OS ID corresponding to the output buffer, outBuf.

Name	I/O	Description
*outPadChar	Input	A pointer to a long value which is the padding character to use, 0 if no padding to be done in the output buffer, outBuf.
*workArea	Input	A character pointer to a 100-byte scratch work area.
*outCharCnt	Input/Output	A pointer to a long value which is the number of characters placed into the output buffer, outBuf.
*outByteCnt	Input/Output	A pointer to a long value which is the number of bytes placed into the output buffer, outBuf.

Return Code

None

Default Behavior

If a suitable translation table is found, the data will be translated from the inCodePage to the outCodePage. If a suitable translate table is not found the data is passed back without translation.

Building on NonStop

The National Language Translation User Exit is built as part of the object library UEXITCO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.

Action	Description
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

TIRXINFO

USEREXIT—NonStop RSC/MP Distributed Processing Flow Data Access Exit

```
Void USEREXIT(short MsgFlow);
```

Source Code

dpsuetdm

Purpose

The USEREXIT server side user exit is called twice for each distributed processing flow and supports multiple APIs. By serving as the main entry point for five APIs, USEREXIT enables inspection and modification of the user data and application data that moves between a Distributed Processing Client (DPC) and a Distributed Processing Server (DPS). USEREXIT is called the first time just after request data has been received from the client. The second call occurs just before the response data is returned to the client.

You can use the set of APIs that USEREXIT supports encrypting and decrypting transaction data, adding custom data that is not generated from CA Gen to the buffer sent to the client side, performing customized auditing, and so on.

Note: The size of the CA Gen data buffer cannot be modified. Data integrity must be maintained by the user exit.

This user exit is available only on the NonStop server platform.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
MsgFlow	Input	Indicates the direction of the data stream available for inspection/modification. Possible values are: 0 - Request message coming from the Client. 1 - Response message going back to the Client.

APIs

USEREXIT is an entry point that supports the use of the following APIs:

Name	Description
GET_MESSAGE_SIZE	Returns the length of the message area for the current request or response message.
GET_USER_DATA	Returns the user data associated with the current request or response message.
SET_USER_DATA	Copies data from the passed in buffer into the user data area associated with the current request or response message.
GET_IEF_DATA	Returns the Gen data (and its length) contained within the current request or response message.
SET_IEF_DATA	Copies data from the passed in buffer into the Gen data area associated with the current request or response message.

Each API supported by USEREXIT is described in the following sections.

Return Code

None

Default Behavior

USEREXIT is called twice for each DPS flow. The user exit is called for the first time just after a request buffer is received from the client. Just before the response buffer is sent back to the client, the user exit is called again.

Building on NonStop

The Distributed Process Flow Data Access User Exit is built as part of the object library DPSUECO. As a prerequisite for building the object library, you must have correct C compiler installed on your system.

Follow these steps:

1. Launch a terminal session
2. Change your current volume to where the IT was installed
3. Run MKEXITS
4. MKEXITS will prompt for confirmation about the source and object code locations used to build the exits. Press enter to accept the default values, otherwise enter new $<volume>.<subvolume> locations at the prompts. After these locations are confirmed, the main menu is displayed. Use this menu to perform the following actions:

Action	Description
X	To extract the exits' source code from their archived file (UEXITSC). Source will be placed into the source location.
E	To invoke the EDIT editor.
T	To invoke the TEDIT editor.
A	To compile all user exits.
R	To link and install compiled user exits into the UEXITCO library
D	Links and installs compiled distributed processing server user exits into the DPSUECO library
P	To access the Peruse facility
O	To change the output location
S	To change the source and object location settings
Q	To exit MKEXITS

Related User Exits

Client Manager/Communications Bridge user exit RSCUserEntry().

API Functions

The APIs described in the following sections can only be called from within the context of the NonStop user exit, USEREXIT().

Function Format

```
short GET_MESSAGE_SIZE( short * msgSize )
```

Purpose

GET_MESSAGE_SIZE returns the size of the current CA Gen message buffer. The reported message buffer size includes the NonStop RSC/MP header, CA Gen data, and user data.

When a MsgFlow parameter of 0 is passed into the main entry point, USEREXIT(), this indicates that the user exit is being called just after the response message is received from the requesting client. The message size returned is that of the message just received from the client.

When a MsgFlow parameter of 1 is passed into the main entry point, this indicates that the user exit is being called just before sending the response message to the requesting client. The message size returned is that of the message about to be sent to the client.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
* msgSize	Output	A pointer to a short where the total size of the current message is stored.

Return Code

The following table gives a brief description of the return code value.

Return Value	Description
Non zero	The size of the CA Gen message buffer.

Default Behavior

If enabled, USEREXIT( ) calls the GET_MESSAGE_SIZE API after receiving a request from the client.

Function Format

```
short GET_USER_DATA( char * data, short length )
```

Purpose

Returns the user data area associated with the current Gen message.

When a MsgFlow parameter of 0 is passed into the main entry point, USEREXIT(), this indicates that the user exit is being called just after the response message is received from the requesting client. The message size returned is that of the message just received from the client.

When a MsgFlow parameter of 1 is passed into the main entry point, this indicates that the user exit is being called just before sending the response message to the requesting client. The message size returned is that of the message about to be sent to the client.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
* data	Input/Output	A pointer to a buffer area where the runtime is to copy the user data. **Note:** It is the caller's responsibility to ensure that adequate memory has been allocated to contain the user data contained in the current message.
Length	Input	The size of the data buffer being passed in.

Return Code

The following table gives a brief description of the return code value.

Return Value	Description
Non zero	The actual number of bytes copied into the data buffer
Zero	Indicates there was no user-data available
Negative	Indicates that the an error occurred

Default Behavior

If enabled, the USEREXIT( ) calls the GET_MESSAGE_DATA API after the request message is received from the client.

Function Format

```
short SET_USER_DATA( char * data, short length )
```

Purpose

Copies the passed in buffer data into the user data area associated with the current Gen message.

If the MsgFlow parameter passed into the main entry point, USEREXIT(), is 0, this signifies that the user exit is being called just after the request message has been received from the client.

If the MsgFlow parameter passed into the main entry point, USEREXIT(), is 1, this signifies that the user exit is being called just before sending the response message to the client. The buffer data will be copied into the current message's user data area before the message is sent to the requesting client.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
* data	Input/Output	A pointer to a buffer area that is copied into the current message's user data area.
length	Input	The length of the data buffer pointed to by the "data" parameter

Return Code

The following table gives a brief description of the return code value.

Return Value	Description
0	Indicates that the data was copied successfully
Non zero	Indicates that the an error occurred

Default Behavior

If enabled, USEREXIT( ) calls the SET_USER_DATA API after the request message is received from the client.

Function Format

```
short GET_IEF_DATA( unsigned char * data, short length )
```

Purpose

Returns a copy of the CA Gen data associated with the current CA Gen message.

If the MsgFlow parameter passed into the main entry point, USEREXIT(), is 0, this signifies that the user exit is being called just after the request message has been received from the client.

If the MsgFlow parameter passed into the main entry point, USEREXIT(), is 1, this signifies that the user exit is being called just before sending the response message to the client. The buffer data will be copied into the current message's user data area before the message is sent to the requesting client.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
* data	Input/Output	A pointer to a buffer area where a copy the Gen data will be written to.
length	Input	The size of the buffer being passed.

Return Code

The following table gives a brief description of the return code value.

Return Value	Description
Non zero	Indicates the actual number of bytes copied into the passed in data buffer.
Negative	Indicates that the an error occurred.

Default Behavior

If enabled, the USEREXIT( ) calls the GET_IEF_DATA API after the request is received from the client.

Function Format

```
short SET_IEF_DATA( unsigned char * data, short length )
```

Purpose

Copies the passed in buffer into the Gen data area associated with the current message.

If the MsgFlow parameter passed into the main entry point, USEREXIT(), is 0, this signifies that the user exit is being called just after the request message has been received from the client.

If the MsgFlow parameter passed into the main entry point, USEREXIT(), is 1, this signifies that the user exit is being called just before sending the response message to the client. The buffer data will be copied into the current message's user data area before the message is sent to the requesting client.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
* data	Input/Output	A pointer to a buffer area that is copied into the current message's Gen data area.
length	Input	The size of the data buffer to be copied.

Return Code

The following table gives a brief description of the return code value.

Return Value	Description
Zero	Indicates the passed in data was successfully copied
Non zero	Indicates that the an error occurred

Default Behavior

If enabled, USEREXIT( ) calls this API after the request is received from the client.

Chapter 6: Web Generation User Exits

User exits are available with the various communication packages used.

Note: For more information about Java user exits used in a distributed processing environment, see the *Distributed Processing – Overview Guide*.

CompareExit—Web Generation Compare Exit

Source Code

CompareExit.java

Purpose

CompareExit is a runtime class used to compare various classes/types with each other.

CompareTo Method—Compares Two Decimals

```
public static int CompareTo(decimal parm1, decimal parm2)
```

Purpose

Compares two decimals and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a decimal to be compared with
parm2	Input	a decimal to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to Java CompareTo() method.

CompareTo Method—Compares Two Characters

```
public static int CompareTo(char parm1, char parm2)
```

Purpose

Compares two chars and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a char to be compared with
parm2	Input	a char to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to Java CompareTo() method.

CompareTo Method—Compares Two Doubles

```
public static int CompareTo(double parm1, double parm2)
```

Purpose

Compares two doubles and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a double to be compared with
parm2	Input	a double to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to Java CompareTo() method.

CompareTo Method—Compares Two Floats

```
public static int CompareTo(float parm1, float parm2)
```

Purpose

Compares two floats and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a float to be compared with
parm2	Input	a float to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to Java CompareTo() method.

CompareTo Method—Compares Two Integers

```
public static int CompareTo(int parm1, int parm2)
```

Purpose

Compares two ints and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	an int to be compared with
parm2	Input	an int to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to Java CompareTo() method.

CompareTo Method—Compares Two Longs

```
public static int CompareTo(long parm1, long parm2)
```

Purpose

Compares two longs and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a long to be compared with
parm2	Input	a long to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to Java CompareTo() method.

CompareTo Method—Compares Two Objects

```
public static int CompareTo(object parm1, object parm2)
```

Purpose

Compares two objects and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

For comparison purposes, nulls are considered equal and a null compared to a non-null is always less than the non-null value.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	an object to be compared with
parm2	Input	an object to be compared to

Return Code

A negative, zero or a positive integer as this object is less than, equal to, or greater than the specified object.

Also this method may throw the InvalidCastException if the object does not implement a 'int CompareTo(Object)' interface like the Comparable interface.

Default Behavior

Depending on the type of parameters, this method delegate the comparison to CompareTo(parm1, parm2) methods defined in this class. If the type of the parameters does not match any of CompareTo(parm1, parm2) method, the method uses .NET Reflection mechanism to find the appropriate CompareTo method for the given types.

CompareTo Method—Compares Two Shorts

```
public static int CompareTo(short parm1, short parm2)
```

Purpose

Compares two shorts and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a short to be compared with
parm2	Input	a short to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to Java CompareTo() method.

CompareTo Method—Compares Two Strings

```
public static int CompareTo(string parm1, string parm2)
```

Purpose

Compares two strings and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

The lengths do not have to be identical, they will still compare as long as characters in the extra length area of the larger string are spaces.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a string to be compared with
parm2	Input	a string to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

The default behavior of this method is as follows.

Check null for both parameters.

Trim the space from strings.

Delegate the comparison to Java CompareTo() method.

CompareTo Method—Compares two strings(upto the indicated length)

```
public static int CompareTo(string parm1, string parm2 , int length)
```

Purpose

Compares two strings and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal. Only the characters up to the indicated length are compared. All characters after that point are ignored.

The lengths do not have to be identical, they will still compare as long as characters in the extra length area of the larger string are spaces.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a string to be compared with
parm2	Input	a string to be compared to
length	Input	an int to indicate the length of the strings to compare

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

The default behavior of this method is as follows.

Check null for both parameters.

Trim the given strings to the specified length.

Delegate the comparison to Java CompareTo() method.

CompareTo Method—Compares Two DateTime instances

```
public static int CompareTo(DateTime parm1, DateTime parm2)
```

Purpose

Compares two DateTime instances and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a DateTime to be compared with
parm2	Input	a DateTime to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to Java CompareTo() method.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

DataConversionExit–Web Generation Data Conversion Exit

Source Code

DataConversionExit.java

Purpose

These exit classes allows users to access data and modify it. This user-exit can be used to modify data saved to and retrieved from a database. It currently handles Strings only. This is sometimes required for some languages where strings stored in a database may need to be presented differently for display. For example DB2 accessed via JDBC can save and retrieve different results from DB2 accessed on the mainframe if the latter is set to an encoding different from that of the JVM.

modifyInputString Method—Modifies Input String

```
public final static String modifyInputString (String parmString)
```

Purpose

It is used to modify strings before they are ultimately saved to a database.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parmString	Input	String to be modified before it is saved.

Return Code

String representing the modified version is returned.

Default Behavior

By default, the strings passed in, are returned unchanged.

modifyOutputString Method—Modifies Output String

```
String modifyOutputString (String parmString)
```

Purpose

It is used to modify strings after they are retrieved from a database.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parmString	Input	String to be modified before it is retrieved.

Return Code

String representing the modified version is returned.

Default Behavior

By default, the strings passed in, are returned unchanged.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

LowerCaseExit– Web Generation Lower Case Exit

Source Code

LowerCaseExit.java

Purpose

LowerCaseExit is a runtime class used to lowercase the given string.

LowerCase Method—Converts String to Lower Case

```
public static string LowerCase(string inStr)
```

Purpose

LowerCase is a used to convert a given string to lowercase, returning a string as well.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
inStr	Input	String to be lowercased.

Return Code

String representing the lower cased version is returned.

Default Behavior

By default, the String.toLowerCase() is used which in turn will call Character.toLowerCase() using default locale for the JVM.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

UpperCaseExit–Web Generation Upper Case Exit

Source Code

UpperCaseExit.java

Purpose

UpperCaseExit is a runtime class used to lowercase the given string.

UpperCase Method—Converts string to Upper Case

```
public static string UpperCase(string inStr)
```

Purpose

UpperCase is a used to convert a given string to uppercase, returning a string as well.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
inStr	Input	a string to be uppercased

Return Code

A string representing the uppercased version of this object.

Default Behavior

By default, the String.toUpperCase() is used which in turn will call Character.toUpperCase() using default locale for the JVM.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

EJBRMIContextExit–Web Generation EJB RMI Context Exit

Source Code

EJBRMIContextExit.java

Purpose

This class will be called from the EJBRMIContext prior to obtaining an InitialContext.

getInstance Method—Retrieves an instance of the exit class

```
EJBRMIContextExit getInstance(Object runtimeObject)
```

Purpose

This method will be called to retrieve an instance of the exit class.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
runtimeobject	Input	Object to be retrieved.

Return Code

Return an instance of the exit class.

Default Behavior

By default, a new instance is created for each request if one does not already exist in the free array.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.

4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

EJBRMIDynamicCoopFlowExit–Web Generation EJB RMI Dynamic Coop Flow Exit

Source Code

EJBRMIDynamicCoopFlowExit.java

Purpose

This class will be called prior to performing an EJBRMI connection from the EJBRMIDynamicCoopFlow. The class will be instantiated with various data and methods will be called to override that data.

getInstance Method—Retrieves an instance of EJBRMIDynamicCoopFlowExit class

```
public static EJBRMIDynamicCoopFlowExit getInstance(String newInitialFactory,
    String newProviderURL,
    String newNextLocation,
    String newProgramID,
    String newTranCode,
    String newProcedureName,
    String newProcedureSourceName,
    String newModelName,
    String newModelShortName,
    String newJavaContext,
    String newJavaPackage,
    Object runtimeObject)
```

Purpose

This method is invoked at the beginning of performing a EJBRMI connection from the EJBRMIDynamicCoopFlow. This method obtains an instance of EJBRMIDynamicCoopFlowExit class and initializes it with the specified parameters.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newInitialFactory	Input	String containing Initial Factory
newProviderURL	Input	String containing Provider URL
newNextLocation	Input	String containing Next Location
newProgramID	Input	String containing Program ID
newTranCode	Input	String containing TranCode
newProcedureName	Input	String containing Procedure Name
newProcedureSourceName	Input	String containing Procedure Source Name
newModelName	Input	String containing Model Name
newModelShortName	Input	String containing Model ShortName
newJavaContext	Input	String containing Java Context
runtimeobject	Input	Object to be retrieved

Return Code

This method returns the initialized EJBRMIDynamicCoopFlowExit object.

Default Behavior

The method uses a simple caching mechanism to obtain a instance of EJBRMIDynamicCoopFlowExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

FreeInstance Method—De-allocates the object obtained with GetInstance()

```
public void FreeInstance()
```

Purpose

At the end of performing a TCPIP connection from the EJBRMIDynamicCoopFlow, This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

ProcessException Method—Indicates whether to retry the operation or to throw an exception

```
public bool ProcessException(int attempts,
    CSUException e)
```

Purpose

This method will be invoked whenever the EJBRMI Coopflow fails to either instantiate the remote object or the server call fails.

Use this exit to indicate whether to retry the operation or to throw an exception.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
attempts	Input	Integer containing the number of attempts
e	Input	CSU Exception

Return Code

The method returns true to retry the flow and false to go ahead and process/throw the exception.

Default Behavior

By default, the method returns false in any case. Also if tracing is enabled, the given exception object is recorded in the trace.

init Method—Initializes the current instance internally from the GetInstance ()

```
public static EJBRMIDynamicCoopFlowExit init(String newInitialFactory,
    String newProviderURL,
    String newNextLocation,
    String newProgramID,
    String newTranCode,
    String newProcedureName,
    String newProcedureSourceName,
    String newModelName,
    String newModelShortName,
    String newJavaContext,
    String newJavaPackage,
    Object runtimeObject)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newInitialFactory	Input	String containing Initial Factory
newProviderURL	Input	String containing Provider URL
newNextLocation	Input	String containing Next Location
newProgramID	Input	String containing Program ID
newTranCode	Input	String containing TranCode
newProcedureName	Input	String containing Procedure Name
newProcedureSourceName	Input	String containing Procedure Source Name
newModelName	Input	String containing Model Name
newModelShortName	Input	String containing Model ShortName
newJavaContext	Input	String containing Java Context
runtimeobject	Input	Object to be retrieved

Return Code

None

Default Behavior

The default behavior of the method is to simply duplicate the specified parameter to the corresponding instance value.

getInitialFactory Method—Retrieve the initial factory classname

```
String getInitialFactory()
```

Purpose

This method will be called to retrieve the initial factory classname to be used for the EJBRMI communications.

Arguments

None

Default Behavior

Return the value set by the constructor.

getProviderURL Method—Retrieves the providerURL

```
String getProviderURL()
```

Purpose

This method will be called to retrieve the providerURL to be used during the EJBRMI communications.

Arguments

None

Default Behavior

Return the value set by the constructor.

getUserObject Method—Retrieves a User Object

```
String getUserObject()
```

Purpose

This method will be called to retrieve a User Object to be passed during the EJBRMI communications.

Arguments

None

Default Behavior

The default value is to return a null.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

EJBRMISecurityExit–Web Generation EJB RMI Security Exit

Source Code

EJBRMISecurityExit.java

Purpose

This class will be called from the EJBRMIContext.

getInstance Method—Allocates a security object that contains all of the security information

```
String getInstance(String user, String pass, String next, String tran, Object object)
```

Purpose

The get Instance method allocates a security object that contains all of the security information that should be passed from the client to the server. This object must include the user runtime object (or enough information to reconstruct it) as well. Because, the security object will be passed in place of the user runtime object in the Request object.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
user	Input	String containing the userID
pass	Input	String containing the password
Next	Input	String containing the next location
Tran	Input	String containing the tranCode
object	Input	runtimeObject

Default Behavior

The method uses a simple caching mechanism to obtain a instance of EJBRMI SecurityExit class and initializes it. Then the method returns the initialized object.

FreeInstance Method—De-allocates the object obtained with GetInstance

```
public void FreeInstance()
```

Purpose

This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

validate Method—Verifies the security object is correct

```
public Object validate(Object object)
```

Purpose

The validate method is used by the server wrapper script to verify that the security object is correct for the server and to extract the user runtime data from the security object.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
object	Input	Security Object

Default Behavior

Always returns a reference to the user runtime data.

getObject Method—Passess the original security object to a Server

```
public String getObject()
```

Purpose

The get Object method is used by a server to pass the original security object when making an EJBRMI flow to another server.

Arguments

None

Default Behavior

By default, the method returns newly instantiated Object class instance.

SecurityType Property—Specifies the type of security

```
public byte SecurityType
```

Purpose

This read-only byte property contains the value to specify what type of security should be used. The valid return values are:

- EJBRMISecurityExit.SECURITY_NO
- EJBRMISecurityExit.SECURITY_STANDARD
- EJBRMISecurityExit.SECURITY_ENHANCED

Default Behavior

The default behavior is returning the value:

EJBRMISecurityExit.SECURITY_NO

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

TCPIPDynamicCoopFlowExit–Web Generation TCPIP Dynamic CoopFlow Exit

Source Code

TCPIPDynamicCoopFlowExit.java

Purpose

The methods in this class will be called prior to performing a TCPIP connection from the TCPIPDynamicCoopFlow. The class will be instantiated with various data and methods will be called to override that data.

getInstance Method—Obtains an instance of TCPIPDynamicCoopFlowExit class

```
public static TCPIPDynamicCoopFlowExit getInstance(String newHostName,
    Integer newPort,
    boolean newClntPersist,
    String newNextLocation,
    String newProgramID,
    String newTranCode,
    String newProcedureName,
    String newProcedureSourceName,
    String newModelName,
    String newModelShortName,
    String newJavaContext,
    String newJavaPackage,
    Object runtimeObject)
```

Purpose

This method is invoked at the beginning of performing a TCPIP connection from the TCPIPDynamicCoopFlow. This method obtains an instance of TCPIPDynamicCoopFlowExit class and initializes it with the specified parameters.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newHostName	Input	String containing HostName

Name	I/O	Description
newPort	Input	String containing Port Number
newClntPersist	Input	Boolean containing Client Persistence
newNextLocation	Input	String containing Next Location
newProgramID	Input	String containing Program ID
newTranCode	Input	String containing TranCode
newProcedureName	Input	String containing Procedure Name
newProcedureSourceName	Input	String containing Procedure Source Name
newModelName	Input	String containing Model Name
newModelShortName	Input	String containing Model ShortName
newJavaContext	Input	String containing Java Context
new JavaPackage	Input	String containing Java Package
Runtimeobject	Input	Object to be retrieved

Return Code

This method returns the initialized TCPIPDynamicCoopFlowExit object.

Default Behavior

The method uses a simple caching mechanism to obtain an instance of TCPIPDynamicCoopFlowExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

FreeInstance Method—De-allocates the object obtained with GetInstance

```
public void FreeInstance()
```

Purpose

At the end of performing a TCPIP connection from the TCPIPDynamicCoopFlow, This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance () invocation.

ProcessException Method—Indicates whether to retry the operation or to throw an exception

```
public bool ProcessException(int attempts,
    CSUException e)
```

Purpose

This method will be invoked whenever the TCPIP Coopflow fails to either instantiate the remote object or the server call fails.

Use this exit to indicate whether to retry the operation or to throw an exception.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
attempts	Input	Integer containing the number of attempts
e	Input	CSU Exception

Return Code

The method returns true to retry the flow and false to go ahead and process/throw the exception.

Default Behavior

By default, the method returns false in any case. Also if tracing is enabled, the given exception object is recorded in the trace.

init Method—Initializes the current instance internally from the GetInstance

```
private void init(String newHostName,
    Integer newPort,
    boolean newClntPersist,
    String newNextLocation,
    String newProgramID,
    String newTranCode,
    String newProcedureName,
    String newProcedureSourceName,
    String newModelName,
    String newModelShortName,
    String newJavaContext,
    String newJavaPackage,
    Object runtimeObject)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newHostName	Input	String containing HostName
newPort	Input	String containing Port Number
newClntPersist	Input	Boolean containing Client Persistence
newNextLocation	Input	String containing Next Location
newProgramID	Input	String containing Program ID
newTranCode	Input	String containing TranCode
newProcedureName	Input	String containing Procedure Name
newProcedureSourceName	Input	String containing Procedure Source Name
newModelName	Input	String containing Model Name
newModelShortName	Input	String containing Model ShortName
newJavaContext	Input	String containing Java Context
new JavaPackage	Input	String containing Java Package

Name	I/O	Description
Runtimeobject	Input	Object to be retrieved

Return Code

None

Default Behavior

The default behavior of the method is to simply duplicate the specified parameter to the corresponding instance value.

getHostName Method—Retrieves the hostname

```
String getHostName()
```

Purpose

This method will be called to retrieve the hostname to be used for the TCPIP communications.

Arguments

None

Default Behavior

Return the value set by the constructor.

getPort Method—Retrieves the port

```
String getPort()
```

Purpose

This method will be called to retrieve the port to be used for the TCPIP communications.

Arguments

None

Default Behavior

Return the value set by the constructor.

geClientPersistence Method—Retrieves the client socket connection persistence state

```
String geClientPersistence()
```

Purpose

This method will be called to retrieve the client socket connection persistence state for the TCPIP communications. A value of "true" indicates a persistent client socket connection. A value of "false" indicates a non persistent client socket connection where the client socket connection is closed after the response is received from the server.

Arguments

None

Default Behavior

Return the value set by the constructor.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

WindowManagerCfgExit–Web Generation Window Manager Configuration

Source Code

WindowManagerCfgExit.java

Purpose

This user exit class is used during Cross-Context Flows. It maps the cross context URL of a load module before flowing or calling a procedure step contained within that load module. The exit allows designers to assign server locations dynamically for load modules of an application. This exit will be called every time a new Procedure Step is accessed, either through Flows or USE PROCEDURE STEP. The URL to the load module which contains the destination Procedure Step is determined by the CA Gen loader. When the URL references a different load module than the current load module, this exit is called to allow overriding the URL mapping location, determined during Assembly of the application. The exit may use any technique (that can be coded in a Java Servlet) to map the passed URL, load module name, and Procedure Step name to another URL.

URL mapURL Method—Maps the passed URL, load module name and procedure step name

```
URL mapURL (URL url, String loadModuleName, String ProcedureStepName)
```

Purpose

Maps the passed URL, load module name and procedure step name to a different URL as needed.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
url	Input	URL
loadModuleName	Input	Name of the Load Module
ProcedureStepName	Input	Name of the Procedure Step

Default Behavior

By default, the URL passed in is returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

ContextLookupExit–Web Generation Context Look Up

Source Code

ContextLookupExit.java

Purpose

This class will be called from the EJBRMIContext prior to obtaining an InitialContext.

lookup Method—Retrieves an instance of the named context object

```
public staticObject lookup (Context ctx, String name)
```

Purpose

This method will be called to retrieve an instance of the named context object.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
ctx	Input	Context
name	Input	Name of the Object

Default Behavior

Returns the named object for the current context.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

CFBDynamicMessageSecurityExit–Web Generation CFB Dynamic Message Security Exit

Source Code

CFBDynamicMessageSecurityExit.java

Purpose

This class will be called from the CFBDynamicMessage.

CFBDynamicMessageSecurityExit Constructor—Provides the default caching mechanism

```
private CFBDynamicMessageSecurityExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The CFBDynamicMessageSecurityExit object.

Default Behavior

This constructor returns The CFBDynamicMessageSecurityExit object.

GetInstance Method—Obtains an instance of CFBDynamicMessageSecurityExit class

```
public static CFBDynamicMessageSecurityExit
    GetInstance (string newUserid,
    string newPassword,
    string newTranCode,
    string newNextLocation, Object runtimeObject)
```

Purpose

This method obtains an instance of CFBDynamicMessageSecurityExit class and initializes it with the private Init().

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newUserid	Input	String containing the new userID
newPassword	Input	String containing the new password
newTranCode	Input	String containing the new tranCode
newNextLocation	Input	String containing the new nextLocation
object	Input	runtimeObject

Return Code

This method returns the initialized CFBDynamicMessageSecurityExit object.

Default Behavior

The method uses a simple caching mechanism to obtain an instance of CFBDynamicMessageSecurityExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

FreeInstance Method—De-allocates the object obtained with GetInstance

```
public void FreeInstance()
```

Purpose

This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

getSecurityToken Method—Allows the user to pass back a security token

```
public byte [] getSecurityToken(int maxLength)
```

Purpose

This method is used to allow the user to pass back a security token to be passed in the enhanced security mode.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
maxLength	Input	Integer parameter indicates the largest byte array that can be returned.

Return Code

The return value is the byte array containing a security token.

Default Behavior

By default a 0 length array is returned.

To indicate an error condition, throw a CSUException. That is, throw new CSUException("CFBDynamicMessageSecurityExit:GetInstance","error message")).

Init Method—Initializes the current instance internally from the GetInstance

```
private void Init(string newUserid,
    string newPassword,
    string newTranCode,
    string newNextLocation, Object runtimeObject)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newUserid	Input	String containing the new userID
newPassword	Input	String containing the new password
newTranCode	Input	String containing the new tranCode
newNextLocation	Input	String containing the new nextLocation
runtimeObject	Input	runtimeObject

Return Code

None

Default Behavior

The default behavior of the method is assigning the specified argument to instance field.

getSecurityType Method—Specifies the type of security

```
public byte getSecurityType
```

Purpose

Returns the value to specify what type of security should be used.

The valid return values are:

CFBDynamicMessageSecurityExit.SECURITY_NO

CFBDynamicMessageSecurityExit.SECURITY_STANDARD

CFBDynamicMessageSecurityExit.SECURITY_ENHANCED

Default Behavior

The default behavior is returning the value:

CFBDynamicMessageSecurityExit.SECURITY_NO

useCMSecurity Method—Specifies whether the Client Manager/Comm Bridge to use the userID and password

```
public bool useCMSecurity
```

Purpose

This returns the value to specify whether the Client Manager/Comm Bridge to use the userID and password values for enhanced security validation or the standard Client Manager/Comm Bridge target server security configuration.

If true is returned, the Client Manager/Comm Bridge uses the userID and password values for enhanced security validation.

If false is returned, the Client Manager/Comm Bridge uses the standard Client Manager/Comm Bridge target server security configuration.

Default Behavior

The default behavior is returning the false.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

CFBDynamicMessageEncodingyExit–Web Generation CFB Dynamic Message Encoding Exit

Source Code

CFBDynamicMessageEncodingExit.java

Purpose

This class will be called from the CFBDynamicMessage.

serverEncoding Method—Retrieves the message text encoding for the named host and transaction

```
public  static String serverEncoding( String tran, String encoding )
```

Purpose

This method will be called to retrieve the message text encoding for the named host and transaction.

The runtimes will always call ServerCodePage to find the correct encoding for each transaction on each server before building the common format message to be sent.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
tran	Input	transaction name associated with this request
encoding	Input	default character encoding for the request

Return Code

This method will be called to retrieve the message text encoding for the named host and transaction.

Default Behavior

The default behavior returns the default encoding. This behavior may be change to return any valid encoding number. The tran argument may be used to select different encodings for each transaction.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

CFBDynamicMessageEncryptionExit–Web Generation CFB Dynamic Message Encryption Exit

Source Code

CFBDynamicMessageEncryptionExit.java

Purpose

This class will be called from the CFBDynamicMessage to encrypt message.

CFBDynamicMessageEncryptionExit Constructor—Provides the default caching mechanism

```
private CFBDynamicMessageEncryptionExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The CFBDynamicMessageEncryptionExit object.

Default Behavior

This constructor returns The CFBDynamicMessageEncryptionExit object.

GetInstance Method—Obtains an instance of CFBDynamicMessageEncryptionExit class and initializes it

```
public static CFBDynamicMessageEncryptionExit
    GetInstance(string newTranCode,
    string newNextLocation,
    string newUserid,Object runtimeObject)
```

Purpose

This method obtains an instance of CFBDynamicMessageEncryptionExit class and initializes it with the private Init().

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newTranCode	Input	String containing the new tranCode
newNextLocation	Input	String containing the new nextLocation
newUserid	Input	String containing the new userID
runtimeObject	Input	Object

Return Code

This method returns the initialized CFBDynamicMessageEncryptionExit object.

Default Behavior

The method uses a simple caching mechanism to obtain an instance of CFBDynamicMessageEncryptionExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

FreeInstance Method—De-allocates the object obtained with GetInstance

```
public void FreeInstance()
```

Purpose

This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

encryptData Method—Allows the user to encrypt the data portion of the message

```
public byte [] encryptData(byte [] data, int maxLength)
```

Purpose

This method is used to allow the user to encrypt the data portion of the message.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
Data	Input	The byte array to be encrypted.
maxLength	Input	Integer parameter indicates the largest byte array that can be returned.

Return Code

The return value is the byte array containing the encrypted data.

Default Behavior

By default the byte array passed in is unchanged and no real encryption occurs. The user should code their own algorithm to encrypt the data.

The byte array passed in can be modified in place and then returned, or if needed a local byte array can be allocated (larger or smaller if needed), populated and returned.

To indicate an error condition, throw a CSUException. That is, throw new CSUException("CFBDynamicMessageEncryptionExit:GetInstance", "error message")).

Init Method—Initializes the current instance internally from the GetInstance

```
private void Init(string newTranCode,
    string newNextLocation,
    string newUserid,object runtimeObject)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newTranCode	Input	String containing the new tranCode
newNextLocation	Input	String containing the new nextLocation
newUserid	Input	String containing the new userID
runtimeObject	Input	Object

Return Code

None

Default Behavior

The default behavior of the method is assigning the specified argument to instance field.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

CFBDynamicMessageDecryptionExit–Web Generation CFB Dynamic Message Decryption

Source Code

CFBDynamicMessageDecryptionExit.java

Purpose

This class will be called after a CFB message has been received from a server.

CFBDynamicMessageDecryptionExit Constructor—Provides the default caching mechanism

```
private CFBDynamicMessageDecryptionExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns the CFBDynamicMessageDecryptionExit object.

Default Behavior

This constructor returns the CFBDynamicMessageDecryptionExit object.

Method

```
public static CFBDynamicMessageDecryptionExit GetInstance(Object runtimeObject)
```

Purpose

This method will be invoked after a CFB message has been received from a server. This method obtains an instance of FBDynamicMessageDecryptionExit class and initializes it with the private Init().

Arguments

The following table gives a brief description of the argument.

Name	I/O	Description
runtimeObject	Input	runtimeObject

Return Code

This method returns the initialized CFBDynamicMessageDecryptionExit object.

Default Behavior

The method uses a simple caching mechanism to obtain an instance of CFBDynamicMessageDecryptionExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

FreeInstance Method—De-allocates the object obtained with GetInstance

```
public void FreeInstance()
```

Purpose

This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

decryptData Method—Decrypts the data portion of the message

```
public byte [] decryptData(byte [] data, int maxLength)
```

Purpose

This method is used to allow the user to decrypt the data portion of the message.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
Data	Input	The byte array to be decrypted.
maxLength	Input	Integer parameter indicates the largest byte array that can be returned.

Return Code

The return value is the byte array containing the decrypted data.

Default Behavior

By default the byte array passed in is unchanged and no real decryption occurs. The user should code their own algorithm to decrypt the data.

The byte array passed in can be modified in place and then returned, or if needed a local byte array can be allocated (larger or smaller if needed), populated and returned.

To indicate an error condition, throw a CSUException. That is, throw new CSUException("CFBDynamicMessageDecryptionExit:GetInstance","error message")).

Init Method—Initializes the current instance internally from the GetInstance

```
private void Init(Object runtimeObject)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of the argument.

Name	I/O	Description
runtimeObject	Input	runtimeObject

Return Code

None

Default Behavior

The default behavior of the method is nothing.

doDecryption Method—Specifies whether decryption should be done

```
byte doDecryption()
```

Arguments

None

Purpose

This method is used to specify whether decryption should be done. The valid return values are: DECRYPTION_NO (default) and DECRYPTION_YES

Default Behavior

By default, DECRYPTION_NO is returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

DefaultYearExit–Web Generation Default Year Exit

Source Code

DefaultYearExit.java

Purpose

This exit class allows the implementation of a customer-specified algorithm addressing Year-2000 concerns.

GetDefaultYear Method—Implements a customer-specified algorithm addressing Year-2000 concerns

```
public static int GetDefaultYear(int inYear, int currentYear)
```

Purpose

GetDefaultYear () is invoked when input editing occurs on a date or timestamp field, and the edit pattern specifies a two character year value. The 4-digit current year and the 2-digit input year are passed to GetDefaultYear (). By default, the current hundred year value is merely added to the two character year value and returned. This method allows the implementation of a customer-specified algorithm addressing Year-2000 concerns.

The default algorithm returns the current hundred years appended with two digit input year. The private method ImputeCenturies() enables an alternative algorithm, implementing a sliding range of hundred year values based on the input decade.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
inYear	Input	Integer containing the 2 or 4 digit year
currentYear	Input	Integer containing the current year

Return Code

Integer containing the four digit year.

Default Behavior

By default, the current hundred-year value is added to the two character year value and returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.

3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.

4. Run JAVAC CompareExit.java.

5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

LocaleExit–Java Locale Exit

Source Code

LocaleExit.java

Purpose

This exit class provides a set of methods that are called at application startup to load customer-specific values for locale editing of displayed application data. Each method provided by this exit is called with the default input, derived from the dialect specified during application design, and should return the appropriate localized value.

getLocalCurrencySymbol Method—Supplies the currency symbol to the generated JAVA application

```
public static char getLocalCurrencySymbol(char def)
```

Purpose

getLocalCurrencySymbol() supplies the currency symbol to the generated JAVA application. The currency symbol is used when editing numeric fields, which includes currency symbol. For example, if the edit pattern is “@ZZZ.ZZZ,99”, the getLocalCurrencySymbol() should specify “@” as the currency symbol.

Arguments

The following table gives a brief description of the argument.

Name	I/O	Description
def	Input	Character containing the default currency symbol

Return Code

Character containing the localized currency symbol.

Default Behavior

By default, Dollar sign ($) is returned.

getLocalThousandsSep Method—Supplies the thousand separator to the generated JAVA application

```
public static char getLocalThousandsSep(char def)
```

Purpose

getLocalThousandsSep () supplies the thousand separator to the generated JAVA application. The thousand separator is used when editing numeric fields, which includes the thousand separator. For example, if the edit pattern is “@ZZZ.ZZZ,99”, the getLocalThousandsSep () should specify “.” as the thousand separator.

Arguments

The following table gives a brief description of the argument.

Name	I/O	Description
def	Input	Character containing the default thousands separator.

Return Code

Character containing the localized thousands separator.

Default Behavior

By default, thousand separator passed in is returned.

getLocalDecimalSeparator Method—Supplies the decimal point to the generated JAVA application

```
public static char getLocalDecimalSeparator(char def)
```

Purpose

getLocalDecimalSep () supplies the decimal point to the generated JAVA application. The decimal point is used when editing numeric fields, which includes decimal point. For example, if the edit pattern is “@ZZZ.ZZZ,99”, the getLocalDecimalSep () should specify “,” as the decimal point.

Arguments

The following table gives a brief description of the argument.

Name	I/O	Description
def	Input	Character containing the default decimal separator

Return Code

Character containing the localized decimal separator.

Default Behavior

By default, decimal separator passed in is returned.

getLocalDateSeparator Method—Supplies the date separator character to the generated JAVA application

```
public static char getLocalDateSeparator(char def)
```

Purpose

getLocalDateSep () supplies the date separator character to the generated JAVA application. The date separator character is used only for date and time fields where the model does not specify the edit pattern. In these cases, the JAVA runtime uses this information to build a default edit pattern using the information provided by the getLocalDateSep(). For example, if the getLocalDateSep() specifies the date separator as "-" (a dash) and the date order is yymmdd, then the default date edit pattern is yy-mm-dd.

Arguments

The following table gives a brief description of the argument.

Name	I/O	Description
def	Input	Character containing the default date separator.

Return Code

Character containing the localized date separator.

Default Behavior

By default, date separator passed in is returned.

getLocalTimeSep Method—Supplies the time separator character to the generated JAVA application

```
public static char getLocalTimeSep(char def)
```

Purpose

getLocalTimeSep () supplies the time separator character to the generated JAVA application. The time separator character is used only for date and time fields where the model does not specify the edit pattern. In these cases, the JAVA runtime uses this information to build a default edit pattern using the information provided by the getLocalTimeSep (). For example, if the getLocalTimeSep () specifies the date separator as ":" (a colon) then the default date edit pattern is yy:mm:dd.

Arguments

The following table gives a brief description of the argument.

Name	I/O	Description
def	Input	Character containing the default time separator.

Return Code

Character containing the localized time separator.

Default Behavior

By default, time separator passed in is returned.

getLocalDateOrder Method—Supplies the date order definition to the generated JAVA application

```
public static char getLocalDateOrder(char def)
```

Purpose

getLocalDateOrder () supplies the date order definition to the generated JAVA application. The date order definition is used only for date and time fields where the model does not specify the edit pattern. In these cases, the JAVA runtime uses this information to build a default edit pattern using the information provided by the getLocalDateOrder (). For example, if the date separator is "-" (a dash) and getLocalDateOrder () specifies the date order as LocaleExit.DATEORDER_YMD or ‘2’, then the default date edit pattern is yy-mm-dd.

Arguments

The following table gives a brief description of the argument.

Name	I/O	Description
def	Input	Character containing the default date order. The following definition or character can be specified. LocaleExit.DATEORDER_MDY or ‘0’ The date order is “MMDDYY”. LocaleExit.DATEORDER_DMY or ‘1’ The date order is “DDMMYY”. LocaleExit.DATEORDER_YDM or ‘2’ The date order is “YYDDMM”.

Return Code

Character containing the localized date order.

Default Behavior

By default, date order passed in is returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.

3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.

4. Run JAVAC CompareExit.java.

5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

RetryLimitExit–Web Generation Retry Limit Exit

Source Code

RetryLimitExit.java

Purpose

This exit class provides configuration of the number of times a procedure step can be retried when the application has requested a “retry transaction”, or when a deadlock condition on the database has been detected.

getUltimateRetryLimit Method—Retrieves the Integer containing absolute upper limit to the number of times a procedure step can be retried

```
public static int getUltimateRetryLimit()
```

Purpose

This method is called by the Java runtime to get the Integer containing absolute upper limit to the number of times a procedure step can be retried. This exit provides a safeguard in case the system attribute "transaction retry limit" is set to an excessive value by an action diagram. This exit defines the upper bound to the retry limit value which can never be exceeded.

Arguments

None

Default Behavior

By default, “99” is returned.

getDefaultRetryLimit Method—Retrieves the Integer containing default retry limit number of times a procedure step can be retried

```
public static int getDefaultRetryLimit()
```

Purpose

This method is called by the Java runtime to get the Integer containing default retry limit to the number of times a Procedure step can be retried in the event that the system attribute "Transaction retry limit" is not set by an action diagram.

Arguments

None

Default Behavior

By default, “10” is returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

SessionIDExit–Web Generation Session ID Exit

Source Code

SessionIDExit.java

Purpose

This exit class is used for CA Gen system attributes.

getSystemId Method—Retrieves the String containing the value for the LOCAL_SYSTEM_ID attributes

```
public static String getSystemId()
```

Purpose

This method is called by the Java runtime to get the String containing the value for the LOCAL_SYSTEM_ID attributes. LOCAL_SYSTEM_ID can be placed on a window during window design. The value can be up to 8 characters in length.

Arguments

None

Default Behavior

By default, “WIN32” is returned.

getUserId Method—Retrieves the String containing the value for the USER_ID attributes

```
public static String getUserId()
```

Purpose

This method is called by the Java runtime to get the String containing the value for the USER_ID attributes. USER_ID can be placed on a window during window design, and referenced by statements in a PrAD. The value can be up to 8 characters in length.

Arguments

None

Default Behavior

By default, “USERID” is returned.

getTerminalId Method—Retrieves the String containing the value for the TERMINAL_ID attributes

```
public static String getTerminalId()
```

Purpose

This method is called by the Java runtime to get the String containing the value for the TERMINAL_ID attributes. TERMINAL_ID can be placed on a window during window design, and referenced by statements in a PrAD. The value can be up to 8 characters in length.

Arguments

None

Default Behavior

By default, “DOMAIN” is returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

SrvrErrorExit–Web Generation Server Error Exit

Source Code

SrvrErrorExit.java

Purpose

This user exit provides methods that are invoked when an error is detected during the processing of a client to server flow.

ServerError Method—Detects an error during the processing of a synchronous client to server flow

public static int ServerError(int failureType, StringBuffer failureCommand, StringBuffer errorList)

Purpose

ServerError() is invoked when an error is detected during the processing of a synchronous client to server flow. The parameter failureType describes the origin of this error and the formatted error message is returned in errorList.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
failureType	Input	An integer value describing the source of the failure. It's value can be one of the following: "CFBUILD" is an error in the construction or parsing of a client/server flow message or response. "XFAL" identifies an error during the server procedures action block execution. "XERR" identifies a communications error occurring somewhere between construction of a message or response, and the deciphering of that message by the partner in this flow.

Name	I/O	Description
failureCommand	Output	This character array can be populated with a command used to reinvoke the failing procedure step. This parameter is only used when returning from the exit with a FailAction of "serverFailedRestart". It will be ignored for any other FailAction.
errorList	Input/Output	An array of characters representing message strings constructed by and normally displayed using the ErrorReport dialog to describe the failure. Each message string is null terminated. Newline characters for formatting are also present as required. The complete list is terminated by more-than-one contiguous null character. On a "serverFailedDisplay" return, errorList, as modified in this exit, will be displayed in the ErrorReport dialog; errorList has a maximum length of 2048 bytes.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
FAILEDDISPLAY	This return value causes the standard error report dialog to be displayed, with return to the previous window.
FAILEDRESTART	This return value suppresses the standard error report and reinvokes the client procedure step that originated the dialog flow. In the case of a failing "procedure step usage," the parent procedure step is returned to at the statement immediately following the Use. For flows designed to return the server's exit state to the client, the exit state set when reinvoking or returning to the client procedure step will be the last value set by the client.
FAILEDTERMINATE	This return value will suppress the standard error report dialog, will not attempt to return to the client procedure step, and will redisplay the client window.

Return Code	Description
FAILEDDISPLAYCUSTOM	This return value will cause a custom error report dialog to be displayed with NO FORMATTING. The dialog returns to the previous window as in previous releases.

Default Behavior

The default return value of FAILEDDISPLAY causes the standard error report dialog to be displayed, with return to the previous client window.

append Method—Formats errors with messages unique to your application

```
public static boolean append(StringBuffer msgOut, int messageNumber, String [] parms,
String dialect)
```

Purpose

This method formats errors with messages unique to your application instead of utilizing Gen error message formatting.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
msgOut	Input/Output	Output buffer to append.
msgNumber	Input/Output	The number of the message in csumessages.properties
Parms	Input/Output	The array of parameters for substitution into the message
Dialect	Input	Specifies the dialect

Return Code

True if appends, false otherwise.

Default Behavior

By default, false is returned.

Method

```
public static boolean append(StringBuffer msgOut, String message)
```

Purpose

This method formats errors with messages unique to your application instead of utilizing CA Gen error message formatting.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
msgOut	Input/Output	Output buffer to append.
message	Input/Output	Message to append.

Return Code

True if appends, false otherwise.

Default Behavior

By default, false is returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

UserExit–Web Generation User Exit

Source Code

UserExit.java

Purpose

UserExit class managing interaction with user customized exits.

Default Behavior

Default Values are:

Name	Value
CurrencySign	LocaleExit.DEF_CURR
ThousandsSeparator	LocaleExit.DEF_THOU
DecimalSeparator	LocaleExit.DEF_DECI
DateSeparator	LocaleExit.DEF_DATE
TimeSeparator	LocaleExit.DEF_TIME
DateOrder	LocaleExit.DEF_ORDER
MessageFile	Uninitialized
DialectName	Uninitialized

startUp Method—Instantiates the UserExit class with its properties initialized

```
startUp (char thousands, char decimal, String message, String dialect)
```

Purpose

This constructor instantiates the UserExit class with its properties initialized.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
thousands	Input	Character that contains the initial thousands separator.
decimal	Input	Character that contains the initial decimal point.
message	Input	Two letter string that contains the initial message file selection key.
dialect	Input	The dialect value that matches the value defined in the model.

Return Code

The UserExit object with the initialized property values based on the given arguments.

Default Behavior

The following values are initialized.

Name	Value
CurrencySign	Return value of invoking LocaleExit.getLocalCurrencySymbol() with the initialized value of “CurrencySign “.
ThousandsSeparator	Return value of invoking LocaleExit.getLocalThousandsSep() with the specified thousands arguments.
DecimalSeparator	Return value of invoking LocaleExit.getLocalDecimalSep() with the specified decimal arguments
DateSeparator	Return value of invoking LocaleExit.getLocalDateSep() with the initialized value of “DateSeparator”.
TimeSeparator	Return value of invoking LocaleExit.getLocalTimeSep() with the initialized value of “TimeSeparator”.
DateOrder	Return value of invoking LocaleExit.getLocalDateOrder() with the initialized value of “DateOrder”.

Name	Value
MessageFile	The specified message argument.
DialectName	The specified Dialect argument with space padded to 8 characters.

getCurrencySign Method—Retrieves the currency sign value for the current UserExit object

```
public char getCurrencySign()
```

Purpose

This method is invoked to get the currency sign value for the current UserExit object.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
str	Input	String that is passed in
length	Input	integer containing the length of the string
padchar	Input	Char containing the pad character

Default Behavior

The default behavior is returning the value initialized.

getThousandsSeparator Method—Retrieves the Thousand Separator value for the current UserExit object

```
public char getThousandsSeparator()
```

Purpose

This method is invoked to get the Thousand Separator value for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized.

getDecimalSeparator Method—Retrieves the Decimal Separator value for the current UserExit object

```
public char getDecimalSeparator()
```

Purpose

This method is invoked to get the Decimal Separator value for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized.

getDateSeparator Method—Retrieves the Date Separator value for the current UserExit object

```
public char getDateSeparator()
```

Purpose

This method is invoked to get the Date Separator value for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized.

getTimeSeparator Method—Retrieves the Time Separator value for the current UserExit object

```
public char getTimeSeparator()
```

Purpose

This method is invoked to get the Time Separator value for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized.

getDateOrder Method—Retrieves the Date Order value for the current UserExit object

```
public char getDateOrder()
```

Purpose

This method is invoked to get the Date Order value for the current UserExit object. The following are the possible values.

Symbol	Value	Description
LocaleExit.DATEORDER_MDY	0	The date order is “MMDDYY”. This value is also specified with LocaleExit.DEF_ORDER.
LocaleExit.DATEORDER_DMY	1	The date order is “DDMMYY”.
LocaleExit.DATEORDER_YDM	2	The date order is “YYDDMM”.

Default Behavior

The default behavior is returning the value initialized.

getMessageFile Method—Retrieves the two letter key to select the message file

```
public string getMessageFile()
```

Purpose

This method is invoked to get the two letter key to select the message file. The following table contains association between key and message file.

Key	Message Filename
WR	ief_Error.js
AR	ief_Error_ar.js
DA	ief_Error_da.js
DU	ief_Error_du.js
FI	ief_Error_fi.js
FR	ief_Error_fr.js
GE	ief_Error_ge.js
HB	ief_Error_hb.js

Key	Message Filename
IT	ief_Error_it.js
JA	ief_Error_ja.js
KO	ief_Error_ko.js
NO	ief_Error_no.js
SP	ief_Error_sp.js
SW	ief_Error_sw.js
<none>	ief_Error.js

Default Behavior

The default behavior is returning the value initialized.

getSystemId Method—Retrieves the system ID string attribute

```
static public string getSystemId()
```

Purpose

This method is invoked to get the system ID string attribute.

Default Behavior

The default behavior is returning the value in the property SessionIdExit.SystemId with its value padded to 8 characters with space.

getUserId Method—Retrieves the userID string attribute

```
static public string getUserId()
```

Purpose

This method is invoked to get the userID string attribute.

Default Behavior

The default behavior is returning the value in the property SessionIdExit.UserId with its value padded to 8 characters with space.

getTerminalId Method—Retrieves the terminal ID string attribute

```
static public string getTerminalId()
```

Purpose

This method is invoked to get the terminal ID string attribute.

Default Behavior

The default behavior is returning the value in the property SessionIdExit. TerminalId with its value is padded to 8 characters with space.

getDialectName Method—Retrieves the current dialect name for the load module

```
static public string getDialectName()
```

Purpose

This method is invoked to get the current dialect name for the load module.

Default Behavior

The default behavior is returning the value initialized.

GetDefaultYear Method—Implements a customer-specified algorithm addressing Year-2000 concerns

```
public static int GetDefaultYear (int  inYear, int currentYear)
```

Purpose

GetDefaultYear () is invoked when input editing occurs on a date or timestamp field, and the edit pattern specifies a 2 character year value. The 4-digit current year and the 2-digit input year are passed to GetDefaultYear (). By default, the current hundred year value is merely added to the 2 character year value and returned. This method allows the implementation of a customer-specified algorithm addressing Year-2000 concerns.

The default algorithm returns the current hundred years appended with two digit input year. The private method ImputeCenturies() enables an alternative algorithm, implementing a sliding range of hundred year values based on the input decade.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
inYear	Input	Integer containing the 2 or 4 digit year.
currentYear	Input	Integer containing the current year.

Return Code

Integer containing the 4 digit year.

Default Behavior

By default, this method invokes DefaultYearExit.GetDefaultYear(inYear, currentYear) with the given argument to this method and the return value is from the invoked method.

padAndTrim Method—Trims and pads the given string with the specified arguments

```
private static String padAndTrim (String str, int length, char padChar)
```

Purpose

This method in used internally to trim and pad the given string with the specified arguments.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
str	Input	String that is passed in
length	Input	integer containing the length of the string
padchar	Input	Char containing the pad characte

Return Code

The passed in string is padded and trimmed and returned.

Default Behavior

The passed in string is padded and trimmed and returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\common subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC CompareExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

WSDynamicCoopFlowExit–Web Service Dynamic Coop Flow Exit

Source Code

WSDynamicCoopFlowExit.java

Purpose

This class is called prior to performing a Web Service connection from the WSDynamicCoopFlow. The class is instantiated with various data.

getInstance Method—Retrieves an Instance of WSDynamicCoopFlowExit Class

```
public static WSDynamicCoopFlowExit getInstance(String newInitialFactory,
String newProviderURL,
String newNextLocation,
String newProgramID,
String newTranCode,
String newProcedureName,
String newProcedureSourceName,
String newModelName,
String newModelShortName,
String newJavaContext,
String newJavaPackage,
Object runtimeObject)
```

Purpose

This method is invoked at the beginning of performing a Web Service connection from the WSDynamicCoopFlow. This method obtains an instance of WSDynamicCoopFlowExit class and initializes it with the specified parameters.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
newBaseURL	Input	String containing Base URL
newContextType	Input	String containing Context Type
newNextLocation	Input	String containing Next Location
newProgramID	Input	String containing Program ID
newTranCode	Input	String containing TranCode
newProcedureName	Input	String containing Procedure Name
newProcedureSource Name	Input	String containing Procedure Source Name

Name	I/O	Description
newModelName	Input	String containing Model Name
newModelShortName	Input	String containing Model ShortName
newJavaContext	Input	String containing Java Context
runtimeobject	Input	Object to be retrieved

Return Code

This method returns the initialized WSDynamicCoopFlowExit object.

Default Behavior

The method uses a simple caching mechanism to obtain an instance of WSDynamicCoopFlowExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

freeInstance Method—De-allocates the Object Obtained with GetInstance()

```
public void freeInstance()
```

Purpose

At the end of performing a Web Service coopflow from the WSDynamicCoopFlow, this method is invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

processException Method—Indicates Whether to Retry the Operation or to Throw an Exception

```
public boolean processException(int attempts,
    CSUException e)
```

Purpose

This method will be invoked whenever an exception has occurred performing a Web Service operation. Use this exit to indicate whether to retry the operation or to throw an exception.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
attempts	Input	Integer containing the number of attempts
E	Input	CSU Exception

Return Code

The method returns true to retry the flow and false to go ahead and process/throw the exception.

Default Behavior

By default, the method returns false in any case. Also if tracing is enabled, the given exception object is recorded in the trace.

init Method—Initializes the Current Instance Internally from the getInstance ()

```
public static WSDynamicCoopFlowExit init(String newInitialFactory,
String newBaseURL,
String newContextType,
String newNextLocation,
String newProgramID,
String newTranCode,
String newProcedureName,
String newProcedureSourceName,
String newModelName,
String newModelShortName,
String newJavaContext,
String newJavaPackage,
Object runtimeObject)
```

Purpose

This private method is invoked internally from the getInstance() to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments:

Name	I/O	Description
newBaseURL	Input	String containing Base URL
newContextType	Input	String containing Context Type
newNextLocation	Input	String containing Next Location
newProgramID	Input	String containing Program ID
newTranCode	Input	String containing TranCode
newProcedureName	Input	String containing Procedure Name
newProcedureSourceName	Input	String containing Procedure Source Name
newModelName	Input	String containing Model Name
newModelShortName		String containing Model ShortName
newJavaContext	Input	String containing Java Context

Name	I/O	Description
runtimeobject	Input	Object to be retrieved

Return Code

None

Default Behavior

The default behavior of the method is to simply assign the specified parameter to the corresponding instance value.

getBaseURL Method—Retrieves the baseURL

```
String getBaseURL()
```

Purpose

This method will be called to retrieve the providerURL to be used during the Web Service communications.

Arguments

None

Default Behavior

Return the value set by the getInstance.

getContextType Method—Retrieves the contextType

```
String getContextType()
```

Purpose

This method will be called to retrieve the contextType to be used during the Web Service communications.

Arguments

None

Default Behavior

Return the value set by the getInstance.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment Java Platform SE 1.6 or higher must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSUxx.JAR.

   Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GENxx\EXITS\coopflow\ws subdirectory.

   Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
4. Run javac WSDynamicCoopFlowExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

Chapter 7: Web View User Exits

WVDefaultYearExit–WebView Default Year Exit

Source Code

WVDefaultYearExit.java

Purpose

This exit class allows the implementation of a customer-specified algorithm addressing Year-2000 concerns.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\WEBVIEW\EXITS subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC WVDefaultYearExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

WVLocaleExit–WebView Locale Exit

Source Code

WVLocaleExit.java

Purpose

This exit class provides a set of methods that are called at application startup to load customer-specific values for locale editing of displayed application data. Each method provided by this exit is called with the default input, derived from the dialect specified during application design, and should return the appropriate localized value.

getLocalCurrencySymbol Method—Supplies the currency symbol to the generated JAVA application

```
public static char getLocalCurrencySymbol(char def)
```

Purpose

getLocalCurrencySymbol() supplies the currency symbol to the generated JAVA application. The currency symbol is used when editing numeric fields, which includes currency symbol. For example, if the edit pattern is “@ZZZ.ZZZ,99”, the getLocalCurrencySymbol() should specify “@” as the currency symbol.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
def	Input	Character containing the default currency symbol.

Return Code

Character containing the localized currency symbol.

Default Behavior

By default, Dollar sign ($) is returned.

getLocalThousandsSep Method—Supplies the thousand separator to the generated JAVA application

```
public static char getLocalThousandsSep(char def)
```

Purpose

getLocalThousandsSep () supplies the thousand separator to the generated JAVA application. The thousand separator is used when editing numeric fields, which includes the thousand separator. For example, if the edit pattern is “@ZZZ.ZZZ,99”, the getLocalThousandsSep () should specify “.” as the thousand separator.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
def	Input	Character containing the default thousands separator.

Return Code

Character containing the localized thousands separator.

Default Behavior

By default, thousand separator passed in is returned.

getLocalDecimalSeparator Method—Supplies the decimal point to the generated JAVA application

```
public static char getLocalDecimalSeparator(char def)
```

Purpose

getLocalDecimalSep () supplies the decimal point to the generated JAVA application. The decimal point is used when editing numeric fields, which includes decimal point. For example, if the edit pattern is “@ZZZ.ZZZ,99”, the getLocalDecimalSep () should specify “,” as the decimal point.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
def	Input	Character containing the default decimal separator.

Return Code

Character containing the localized decimal separator.

Default Behavior

By default, decimal separator passed in is returned.

getLocalDateSeparator Method—Supplies the date separator character to the generated JAVA application

```
public static char getLocalDateSeparator(char def)
```

Purpose

getLocalDateSep () supplies the date separator character to the generated JAVA application. The date separator character is used only for date and time fields where the model does not specify the edit pattern. In these cases, the JAVA runtime uses this information to build a default edit pattern using the information provided by the getLocalDateSep(). For example, if the getLocalDateSep() specifies the date separator as "-" (a dash) and the date order is yymmdd, then the default date edit pattern is yy-mm-dd.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
def	Input	Character containing the default date separator.

Return Code

Character containing the localized date separator.

Default Behavior

By default, date separator passed in is returned.

getLocalTimeSep Method—Supplies the time separator character to the generated JAVA application

```
public static char getLocalTimeSep(char def)
```

Purpose

getLocalTimeSep () supplies the time separator character to the generated JAVA application. The time separator character is used only for date and time fields where the model does not specify the edit pattern. In these cases, the JAVA runtime uses this information to build a default edit pattern using the information provided by the getLocalTimeSep (). For example, if the getLocalTimeSep () specifies the date separator as ":" (a colon) then the default date edit pattern is yy:mm:dd.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
def	Input	Character containing the default time separator.

Return Code

Character containing the localized time separator.

Default Behavior

By default, time separator passed in is returned.

getLocalDateOrder Method—Supplies the date order definition to the generated JAVA application

```
public static char getLocalDateOrder(char def)
```

Purpose

getLocalDateOrder () supplies the date order definition to the generated JAVA application. The date order definition is used only for date and time fields where the model does not specify the edit pattern. In these cases, the JAVA runtime uses this information to build a default edit pattern using the information provided by the getLocalDateOrder (). For example, if the date separator is "-" (a dash) and getLocalDateOrder () specifies the date order as LocaleExit.DATEORDER_YMD or ‘2’, then the default date edit pattern is yy-mm-dd.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
def	Input	Character containing the default date order. The following definition or character can be specified. LocaleExit.DATEORDER_MDY or ‘0’ The date order is “MMDDYY”. LocaleExit.DATEORDER_DMY or ‘1’ The date order is “DDMMYY”. LocaleExit.DATEORDER_YDM or ‘2’ The date order is “YYDDMM”.

Return Code

Character containing the localized date order.

Default Behavior

By default, date order passed in is returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\WEBVIEW\EXITS subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC WVDefaultYearExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

WVRetryLimitExit–WebView Retry Limit Exit

Source Code

WVRetryLimitExit.java

Purpose

This exit class provides configuration of the number of times a procedure step can be retried when the application has requested a “retry transaction”, or when a deadlock condition on the database has been detected.

getUltimateRetryLimit Method—Retrieves the Integer containing absolute upper limit

```
public static int getUltimateRetryLimit()
```

Purpose

This method is called by the Java runtime to get the Integer containing absolute upper limit to the number of times a procedure step can be retried. This exit provides a safeguard in case the system attribute "transaction retry limit" is set to an excessive value by an action diagram. This exit defines the upper bound to the retry limit value which can never be exceeded.

Arguments

None

Default Behavior

By default, “99” is returned.

getDefaultRetryLimit Method—Retrieves the Integer containing default retry limit

```
public static int getDefaultRetryLimit()
```

Purpose

This method is called by the Java runtime to get the Integer containing default retry limit to the number of times a Procedure step can be retried in the event that the system attribute "Transaction retry limit" is not set by an action diagram.

Arguments

None

Default Behavior

By default, “10” is returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\WEBVIEW\EXITS subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC WVDefaultYearExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

WVSessionIDExit–WebView Session ID Exit

Source Code

WVSessionIDExit.java

Purpose

This exit class is used for CA Gen system attributes.

getSystemId Method—Retrieves the String containing the value for the LOCAL_SYSTEM_ID attributes

```
public static String getSystemId()
```

Purpose

This method is called by the Java runtime to get the String containing the value for the LOCAL_SYSTEM_ID attributes. LOCAL_SYSTEM_ID can be placed on a window during window design. The value can be up to 8 characters in length.

Arguments

None

Default Behavior

By default, “WIN32” is returned.

getUserId Method—Retrieves the String containing the value for the USER_ID attributes

```
public static String getUserId()
```

Purpose

This method is called by the Java runtime to get the String containing the value for the USER_ID attributes. USER_ID can be placed on a window during window design, and referenced by statements in a PrAD. The value can be up to 8 characters in length.

Arguments

None

Default Behavior

By default, “USERID” is returned.

getTerminalId Method—Retrieves the String containing the value for the TERMINAL_ID attributes

```
public static String getTerminalId()
```

Purpose

This method is called by the Java runtime to get the String containing the value for the TERMINAL_ID attributes. TERMINAL_ID can be placed on a window during window design, and referenced by statements in a PrAD. The value can be up to 8 characters in length.

Arguments

None

Default Behavior

By default, “DOMAIN” is returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\WEBVIEW\EXITS subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC WVDefaultYearExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

WVSrvrErrorExit–WebView Server Error Exit

Source Code

WVSrvrErrorExit.java

Purpose

This user exit provides methods that are invoked when an error is detected during the processing of a client to server flow.

ServerError Method—Detects an error during the processing of a synchronous client to server flow

```
public static int int ServerError(int failureType, StringBuffer errorList)
```

Purpose

ServerError() is invoked when an error is detected during the processing of a synchronous client to server flow. The parameter failureType describes the origin of this error and the formatted error message is returned in errorList.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
failureType	Input	An integer value describing the source of the failure. It's value can be one of the following: "CFBUILD" is an error in the construction or parsing of a client/server flow message or response. "XFAL" identifies an error during the server procedures action block execution. "XERR" identifies a communications error occurring somewhere between construction of a message or response, and the deciphering of that message by the partner in this flow.
errorList	Input/Output	An array of characters representing message strings constructed by and normally displayed using the ErrorReport dialog to describe the failure. Each message string is null terminated. Newline characters for formatting are also present as required. The complete list is terminated by more-than-one contiguous null character. On a "serverFailedDisplay" return, errorList, as modified in this exit, will be displayed in the ErrorReport dialog; errorList has a maximum length of 2048 bytes.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
FAILEDDISPLAY	This return value causes the standard error report dialog to be displayed, with return to the previous window.
FAILEDRESTART	This return value suppresses the standard error report and reinvokes the client procedure step that originated the dialog flow. In the case of a failing "procedure step usage," the parent procedure step is returned to at the statement immediately following the Use. For flows designed to return the server's exit state to the client, the exit state set when reinvoking or returning to the client procedure step will be the last value set by the client.
FAILEDTERMINATE	This return value will suppress the standard error report dialog, will not attempt to return to the client procedure step, and will redisplay the client window.
FAILEDDISPLAYCUSTOM	This return value will cause a custom error report dialog to be displayed with NO FORMATTING. The dialog returns to the previous window as in previous releases.

Default Behavior

The default return value of FAILEDDISPLAY causes the standard error report dialog to be displayed, with return to the previous client window.

Method

```
public static boolean append(StringBuffer msgOut, int messageNumber, String [] parms,
String dialect)
```

Purpose

This method formats errors with messages unique to your application instead of utilizing Gen error message formatting.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
msgOut	Input/Output	Output buffer to append.
msgNumber	Input/Output	The number of the message in csumessages.properties
Parms	Input/Output	The array of parameters for substitution into the message
Dialect	Input	Specifies the dialect

Return Code

True if appends, false otherwise.

Default Behavior

By default, false is returned.

Method

```
public static boolean append(StringBuffer msgOut, String message)
```

Purpose

This method formats errors with messages unique to your application instead of utilizing CA Gen error message formatting.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
msgOut	Input/Output	Output buffer to append.
message	Input/Output	Message to append.

Return Code

True if appends, false otherwise.

Default Behavior

By default, false is returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\WEBVIEW\EXITS subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC WVDefaultYearExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

WVUserExit–WebView User Exit

Source Code

WVUserExit.java

Purpose

UserExit class managing interaction with user customized exits.

Default Behavior

Default Values are:

Name	Value
CurrencySign	LocaleExit.DEF_CURR

Name	Value
ThousandsSeparator	LocaleExit.DEF_THOU
DecimalSeparator	LocaleExit.DEF_DECI
DateSeparator	LocaleExit.DEF_DATE
TimeSeparator	LocaleExit.DEF_TIME
DateOrder	LocaleExit.DEF_ORDER
MessageFile	Uninitialized
DialectName	Uninitialized

startUp Method—Instantiates the UserExit class with its properties initialized

```
startUp (char thousands, char decimal, String message, String dialect)
```

Purpose

This constructor instantiates the UserExit class with its properties initialized.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
thousands	Input	Character that contains the initial thousands separator.
decimal	Input	Character that contains the initial decimal point.
message	Input	Two letter string that contains the initial message file selection key.
dialect	Input	The dialect value that matches the value defined in the model.

Return Code

The UserExit object with the initialized property values based on the given arguments.

Default Behavior

The following values are initialized.

Name	Value
CurrencySign	Return value of invoking LocaleExit.getLocalCurrencySymbol() with the initialized value of “CurrencySign “.
ThousandsSeparator	Return value of invoking LocaleExit.getLocalThousandsSep() with the specified thousands arguments.
DecimalSeparator	Return value of invoking LocaleExit.getLocalDecimalSep() with the specified decimal arguments
DateSeparator	Return value of invoking LocaleExit.getLocalDateSep() with the initialized value of “DateSeparator”.
TimeSeparator	Return value of invoking LocaleExit.getLocalTimeSep() with the initialized value of “TimeSeparator”.
DateOrder	Return value of invoking LocaleExit.getLocalDateOrder() with the initialized value of “DateOrder”.
MessageFile	The specified message argument.
DialectName	The specified Dialect argument with space padded to 8 characters.

getCurrencySign Method—Retrieves the currency sign value for the current UserExit object

```
public char getCurrencySign()
```

Purpose

This method is invoked to get the currency sign value for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized.

getThousandsSeparator Method—Retrieves the Thousand Separator value for the current UserExit object

```
public char getThousandsSeparator()
```

Purpose

This method is invoked to get the Thousand Separator value for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized.

getDecimalSeparator Method—Retrieves the Decimal Separator value for the current UserExit object

```
public char getDecimalSeparator()
```

Purpose

This method is invoked to get the Decimal Separator value for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized.

getDateSeparator Method—Retrieves the Date Separator value for the current UserExit object

```
public char getDateSeparator()
```

Purpose

This method is invoked to get the Date Separator value for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized.

getTimeSeparator Method—Retrieves the Time Separator value for the current UserExit object

```
public char getTimeSeparator()
```

Purpose

This method is invoked to get the Time Separator value for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized.

getDateOrder Method—Retrieves the Date Order value for the current UserExit object

```
public char getDateOrder()
```

Purpose

This method is invoked to get the Date Order value for the current UserExit object. The following are the possible values.

Symbol	Value	Description
LocaleExit.DATEORDER_MDY	0	The date order is “MMDDYY”. This value is also specified with LocaleExit.DEF_ORDER.
LocaleExit.DATEORDER_DMY	1	The date order is “DDMMYY”.
LocaleExit.DATEORDER_YDM	2	The date order is “YYDDMM”.

Default Behavior

The default behavior is returning the value initialized.

getMessageFile Method—Retrieves the two letter key to select the message file

```
public string getMessageFile()
```

Purpose

This method is invoked to get the two letter key to select the message file. The following table contains association between key and message file.

Key	Message Filename
WR	ief_Error.js
AR	ief_Error_ar.js
DA	ief_Error_da.js
DU	ief_Error_du.js
FI	ief_Error_fi.js
FR	ief_Error_fr.js
GE	ief_Error_ge.js
HB	ief_Error_hb.js
IT	ief_Error_it.js
JA	ief_Error_ja.js
KO	ief_Error_ko.js
NO	ief_Error_no.js
SP	ief_Error_sp.js
SW	ief_Error_sw.js
<none>	ief_Error.js

Default Behavior

The default behavior is returning the value initialized.

getSystemId Method—Retrieves the system ID string attribute

```
static public string getSystemId()
```

Purpose

This method is invoked to get the system ID string attribute.

Default Behavior

The default behavior is returning the value in the property SessionIdExit.SystemId with its value padded to 8 characters with space.

getUserId Method—Retrieves the user ID string attribute

```
static public string getUserId()
```

Purpose

This method is invoked to get the userID string attribute.

Default Behavior

The default behavior is returning the value in the property SessionIdExit.UserId with its value padded to 8 characters with space.

getTerminalId Method—Retrieves the terminal ID string attribute

```
static public string getTerminalId()
```

Purpose

This method is invoked to get the terminal ID string attribute.

Default Behavior

The default behavior is returning the value in the property SessionIdExit. TerminalId with its value is padded to 8 characters with space.

getDialectName Method—Retrieves the current dialect name for the load module

```
static public string getDialectName()
```

Purpose

This method is invoked to get the current dialect name for the load module.

Default Behavior

The default behavior is returning the value initialized.

GetDefaultYear Method—Implements of a customer-specified algorithm addressing Year-2000 concerns

```
public static int GetDefaultYear (int  inYear, int currentYear)
```

Purpose

GetDefaultYear () is invoked when input editing occurs on a date or timestamp field, and the edit pattern specifies a 2 character year value. The 4-digit current year and the 2-digit input year are passed to GetDefaultYear (). By default, the current hundred year value is merely added to the 2 character year value and returned. This method allows the implementation of a customer-specified algorithm addressing Year-2000 concerns.

The default algorithm returns the current hundred years appended with two digit input year. The private method ImputeCenturies() enables an alternative algorithm, implementing a sliding range of hundred year values based on the input decade.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
inYear	Input	Integer containing the 2 or 4 digit year.
currentYear	Input	Integer containing the current year.

Return Code

Integer containing the 4 digit year.

Default Behavior

By default, this method invokes DefaultYearExit.GetDefaultYear(inYear, currentYear) with the given argument to this method and the return value is from the invoked method.

padAndTrim Method—Trims and pads the given string with the specified arguments

```
private static String padAndTrim (String str, int length, char padChar)
```

Purpose

This method in used internally to trim and pad the given string with the specified arguments.

Arguments

The following table gives a brief description of the arguments.

Name	I/O	Description
str	Input	String that is passed in
length	Input	Integer containing the length of the string
padchar	Input	Char containing the pad character

Return Code

The passed in string is padded and trimmed and returned.

Default Behavior

The passed in string is padded and trimmed and returned.

Rebuilding the Exit

This exit is compiled into a Java class file. The Java development environment J2SE 1.6 or greater must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSU*xx*.JAR, where *xx* is the CA Gen release number.
3. Change your current directory to that which contains the java source file.

   Typically this will be in the CA Gen install area, within the CLASSES\COM\CA\GEN*xx*\EXITS\WEBVIEW\EXITS subdirectory, where *xx* is the current CA Gen release number.
4. Run JAVAC WVDefaultYearExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

Chapter 8: .NET User Exits

ASP.NET Web Client User Exits

This section describes the user exits that are located in <CAGen-root>\.net\exits\src\amrt directory and used by the CA Gen ASP.NET Web Clients.

com.ca.gen.exits.amrt.DefaultYearExit–C# Default Year Exit

Source Code

DefaultYearExit.cs

Purpose

This exit class allows the implementation of a customer-specified algorithm addressing Year-2000 concerns.

Method

```
public static int GetDefaultYear(int inYear, int currentYear)
```

Purpose

GetDefaultYear () is invoked when input editing occurs on a date or timestamp field, and the edit pattern specifies a 2 character year value. The 4-digit current year and the 2-digit input year are passed to GetDefaultYear (). By default, the current hundred year value is merely added to the 2 character year value and returned. This method allows the implementation of a customer-specified algorithm addressing Year-2000 concerns.

The default algorithm returns the current hundred years appended with two digit input year. The private method ImputeCenturies() enables an alternative algorithm, implementing a sliding range of hundred year values based on the input decade.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
inYear	Input	Integer containing the 2 or 4 digit year.

Name	I/O	Description
currentYear	Input	Integer containing the current year.

Return Code

Integer containing the 4 digit year.

Default Behavior

By default, the current hundred-year value is added to the 2-character year value and returned.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.amrt.LocaleExit–C# Locale Exit

Source Code

LocaleExit.cs

Purpose

This exit class provides a set of methods that are called at startup of application to load customer-specific values for locale editing of data displayed in application. Each method provided by this exit is called with the default input, derived from the dialect specified during application design, and should return the appropriate localized value.

Method

```
public static char getLocalCurrencySymbol(char def)
```

Purpose

getLocalCurrencySymbol() supplies the currency symbol to the generated .NET application. The currency symbol is used when editing numeric fields, which includes currency symbol. For example, if the edit pattern is “@ZZZ.ZZZ,99”, the getLocalCurrencySymbol() should specify “@” as the currency symbol.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
def	Input	Character containing the default currency symbol.

Return Code

Character containing the localized currency symbol.

Default Behavior

By default, dollar sign ‘$’ is returned.

Method

```
public static char getLocalThousandsSep(char def)
```

Purpose

getLocalThousandsSep () supplies the thousand separator to the generated .NET application. The thousand separator is used when editing numeric fields, which includes the thousand separator. For example, if the edit pattern is “@ZZZ.ZZZ,99”, the getLocalThousandsSep () should specify “.” as the thousand separator.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
def	Input	Character containing the default thousands separator.

Return Code

Character containing the localized thousands separator.

Default Behavior

By default, thousand separator passed in is returned.

Method

```
public static char getLocalDecimalSep(char def)
```

Purpose

getLocalDecimalSep () supplies the decimal point to the generated .NET application. The decimal point is used when editing numeric fields, which includes decimal point. For example, if the edit pattern is “@ZZZ.ZZZ,99”, the getLocalDecimalSep () should specify “,” as the decimal point.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
def	Input	Character containing the default decimal separator.

Return Code

Character containing the localized decimal point.

Default Behavior

By default, decimal point passed in is returned.

Method

```
public static char getLocalDateSep(char def)
```

Purpose

getLocalDateSep () supplies the date separator character to the generated .NET application. The date separator character is used only for date and time fields where the model does not specify the edit pattern. In these cases, the .NET runtime uses this information to build a default edit pattern using the information provided by the getLocalDateSep(). For example, if the getLocalDateSep() specifies the date separator as "-" (a dash) and the date order is yymmdd, then the default date edit pattern is yy-mm-dd.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
def	Input	Character containing the default date separator.

Return Code

Character containing the localized date separator.

Default Behavior

By default, date separator passed in is returned.

Method

```
public static char getLocalTimeSep(char def)
```

Purpose

getLocalTimeSep () supplies the time separator character to the generated .NET application. The time separator character is used only for date and time fields where the model does not specify the edit pattern. In these cases, the .NET runtime uses this information to build a default edit pattern using the information provided by the getLocalTimeSep (). For example, if the getLocalTimeSep () specifies the date separator as ":" (a colon) then the default date edit pattern is yy:mm:dd.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
def	Input	Character containing the default time separator.

Return Code

Character containing the localized time separator.

Default Behavior

By default, time separator passed in is returned.

Method

```
public static char getLocalDateOrder(char def)
```

Purpose

getLocalDateOrder () supplies the date order definition to the generated .NET application. The date order definition is used only for date and time fields where the model does not specify the edit pattern. In these cases, the .NET runtime uses this information to build a default edit pattern using the information provided by the getLocalDateOrder (). For example, if the date separator is "-" (a dash) and getLocalDateOrder () specifies the date order as LocaleExit.DATEORDER_YMD or ‘2’, then the default date edit pattern is yy-mm-dd.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
def	Input	Character containing the default date order. The following definition or character can be specified. LocaleExit.DATEORDER_MDY or ‘0’ The date order is “MMDDYY”. LocaleExit.DATEORDER_DMY or ‘1’ The date order is “DDMMYY”. LocaleExit.DATEORDER_YDM or ‘2’ The date order is “YYDDMM”.

Return Code

Character containing the localized date order.

Default Behavior

By default, date order passed in is returned.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.amrt.RetryLimitExit–C# Retry Limit Exit

Source Code

RetryLimitExit.cs

Purpose

This exit class provides configuration of the number of times a procedure step can be retried when the application has requested a "retry transaction", or when a deadlock condition on the database has been detected.

Property

```
public static int UltimateRetryLimit
```

Purpose

This property is a Integer containing absolute upper limit to the number of times a procedure step can be retried. This exit provides a safeguard in case the system attribute "transaction retry limit" is set to an excessive value by an action diagram. This exit defines the upper bound to the retry limit value which can never be exceeded.

Default Value

By default, "99" is returned.

Property

```
public static int DefaultRetryLimit
```

Purpose

This property is Integer containing default retry limit to the number of times a Procedure step can be retried in the event that the system attribute "Transaction retry limit" is not set by an action diagram.

Default Value

By default, “10” is returned.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with ‘Run as administrator’.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.amrt.SessionIdExit–C# Session ID Exit

Source Code

SessionIdExit.cs

Purpose

This exit class is used for CA Gen system attributes.

Property

```
public static string SystemId
```

Purpose

This property is string containing the value for the LOCAL_SYSTEM_ID attributes. LOCAL_SYSTEM_ID can be placed on a window during window design. The value can be up to 8 characters in length.

Default Value

By default, “WIN32” is returned.

Property

```
public static string UserId
```

Return Code

This property is string containing the value for the USER_ID attributes. USER_ID can be placed on a window during window design, and referenced by statements in a PrAD. The value can be up to 8 characters in length.

Default Value

By default, “USERID” is returned.

Property

```
public static string TerminalId
```

Purpose

This property is string containing the value for the TERMINAL_ID attributes. TERMINAL_ID can be placed on a window during window design, and referenced by statements in a PrAD. The value can be up to 8 characters in length.

Default Value

By default, “DOMAIN” is returned.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.amrt.SrvrErrorExit–C# Server Error Exit

Source Code

SrvrErrorExit.cs

Purpose

This exit class provides methods that are invoked when an error is detected during the processing of a client to server flow.

Method

```
public static int ServerError(int failureType,
        StringBuilder errorList)
```

Purpose

ServerError() is invoked when an error is detected during the processing of a synchronous client to server flow. The parameter failureType describes the origin of this error and the formatted error message is returned in errorList.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
failureType	Input	An integer value describing the source of the failure. It's value can be one of the following: com.ca.gen.fmrt. ExternalLoader.CFBUILD or ‘0’: An error in the construction or parsing of a client/server flow message or response. com.ca.gen.fmrt. ExternalLoader.XFAL or ‘1’: An error during the server procedures action block execution. com.ca.gen.fmrt. ExternalLoader.XERR or ‘2’: A communications error occurring somewhere between construction of a message or response, and the deciphering of that message by the partner in this flow.
errorList	Input/Output	A mutable string, StringBuilder, representing message strings constructed by and normally displayed using the ErrorReport dialog to describe the failure. Each message string is null terminated. Newline characters for formatting are also present as required. On a "serverFailedDisplay" return, errorList, as modified in this exit, will be displayed in the ErrorReport dialog.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
SrvrErrorExit.FAILEDDISPLAY	This return value causes the standard error report dialog to be displayed, with return to the previous window. The integer value is ‘3’.

Return Code	Description
SrvrErrorExit.FAILEDRESTART	This return value suppresses the standard error report and re-invokes the client procedure step that originated the dialog flow. In the case of a failing "procedure step usage," the parent procedure step is returned to at the statement immediately following the Use. In both cases, if failureCommand is set it is used as the system command when re-invoking or returning to the client procedure step. For flows designed to return the server's exit state to the client, the exit state set when re-invoking or returning to the client procedure step will be the last value set by the client. The integer value is ‘1’.
SrvrErrorExit.FAILEDTERMINATE	This return value will suppress the standard error report dialog, will not attempt to return to the client procedure step, and will redisplay the client window. The integer value is ‘2’.
SrvrErrorExit.FAILEDDISPLAYCUSTOM	This return value will causes a custom error report dialog to be displayed with NO FORMATTING. The dialog returns to the previous window as in previous releases. The integer value is ‘4’.

Default Behavior

The default return value of SrvrErrorExit.FAILEDDISPLAY causes the standard error report dialog to be displayed, with return to the previous client window.

Method

```
public static bool Append(StringBuilder msgOut,
        int messageNumber,
        string [] parms,
        string dialect)
```

Purpose

This method formats errors with messages unique to your application instead of utilizing Gen error message formatting.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
msgOut	Input/Output	Output buffer to append
messageNumber	Input/Output	The number of the message in csumessages.properties
parms	Input/Output	The array of parameters for substitution into the message
dialect	Input	Specifies the dialect

Return Code

True if appends, false otherwise.

Default Behavior

By default, false is returned.

Method

```
public static bool Append(StringBuilder msgOut, string message)
```

Purpose

This method formats errors with messages unique to your application instead of utilizing CA Gen error message formatting.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
msgOut	Input/Output	Output buffer to append.
message	Input/Output	Message to append.

Return Code

True if appends, false otherwise.

Default Behavior

By default, false is returned.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.amrt.UserExit–C# User Exit

Source Code

UserExit.cs

Purpose

This class manages interaction with various user customized exits through an instantiated object.

Constructor

```
public UserExit()
```

Purpose

This constructor is invoked as a part of a UserExit class instantiation to initialize properties to its base default value.

Arguments

None

Return Code

The UserExit object initialized with the default property values.

Default Behavior

This constructor initializes the properties as follows.

Property	Value
CurrencySign	LocaleExit.DEF_CURR
ThousandsSeparator	LocaleExit.DEF_THOU
DecimalSeparator	LocaleExit.DEF_DECI
DateSeparator	LocaleExit.DEF_DATE
TimeSeparator	LocaleExit.DEF_TIME
DateOrder	LocaleExit.DEF_ORDER
MessageFile	Uninitialized
DialectName	Uninitialized

Constructor

```
public UserExit(char thousands,
        char decimall,
        string message,
        string dialect)
```

Purpose

This constructor instantiates the UserExit class with its properties initialized.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
thousands	Input	Character that contains the initial thousands separator.
decimall	Input	Character that contains the initial decimal point.
message	Input	Two letter string that contains the initial message file selection key.
dialect	Input	The dialect value that matches the value defined in the model.

Return Code

The UserExit object with the initialized property values based on the given arguments.

Default Behavior

This constructor initializes the properties as follows.

Property	Value
CurrencySign	Return value of invoking LocaleExit.getLocalCurrencySymbol() with _CurrencySign field that is initialized with UserExit() constructor.
ThousandsSeparator	Return value of invoking LocaleExit.getLocalThousandsSep() with the specified thousands arguments.
DecimalSeparator	Return value of invoking LocaleExit.getLocalDecimalSep() with the specified decimall arguments
DateSeparator	Return value of invoking LocaleExit.getLocalDateSep() with _DateSeparator field that is initialized with UserExit() constructor.
TimeSeparator	Return value of invoking LocaleExit.getLocalDateSep() with _TimeSeparator field that is initialized with UserExit() constructor.
DateOrder	Return value of invoking LocaleExit.getLocalDateSep() with _DateOrder field that is initialized with UserExit() constructor.
MessageFile	The specified message argument.
DialectName	The specified Dialect argument with space padded to 8 characters.

Property

```
public char CurrencySign
```

Purpose

This read-only property contains the currency sign for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized in constructor. For more detail, please see the constructor section above.

Property

```
public char ThousandsSeparator
```

Purpose

This read-only property contains the thousands separator for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized in constructor. For more detail, please see the constructor section above.

Property

public char DecimalSeparator

Purpose

This read-only property contains the Decimal Point for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized in constructor. For more detail, please see the constructor section above.

Property

```
public char DateSeparator
```

Purpose

This read-only property contains the Date Separator for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized in constructor. For more detail, please see the constructor section above.

Property

```
public char TimeSeparator
```

Purpose

This read-only property contains the Time Separator for the current UserExit object.

Default Behavior

The default behavior is returning the value initialized in constructor. For more detail, please see the constructor section above.

Property

```
public char DateOrder
```

Purpose

This read-only property contains the Date Order for the current UserExit object. The following is the possible value for this property.

Symbol	Value	Description
LocaleExit.DATEORDER_MDY	0	The date order is “MMDDYY”. This value is also specified with LocaleExit.DEF_ORDER.
LocaleExit.DATEORDER_DMY	1	The date order is “DDMMYY”.
LocaleExit.DATEORDER_YDM	2	The date order is “YYDDMM”.

Default Behavior

The default behavior is returning the value initialized in constructor. For more detail, please see the constructor section above.

Property

```
public string MessageFile
```

Purpose

This read-only property contains the two letter key to select the message file. The following table contains association between key and message file.

Key	Message Filename
WR	ief_Error.js
AR	ief_Error_ar.js

Key	Message Filename
DA	ief_Error_da.js
DU	ief_Error_du.js
FI	ief_Error_fi.js
FR	ief_Error_fr.js
GE	ief_Error_ge.js
HB	ief_Error_hb.js
IT	ief_Error_it.js
JA	ief_Error_ja.js
KO	ief_Error_ko.js
NO	ief_Error_no.js
SP	ief_Error_sp.js
SW	ief_Error_sw.js
<none>	ief_Error.js

Default Behavior

The default behavior is returning the value initialized in constructor. For more detail, please see the constructor section above.

Property

```
static public string SystemId
```

Purpose

This read-only property provides the system ID string attribute.

Default Behavior

The default behavior is returning the value in the property SessionIdExit.SystemId with its value is padded to 8 characters with space.

Property

```
static public string UserId
```

Purpose

This read-only property provides the user ID string attribute.

Default Behavior

The default behavior is returning the value in the property SessionIdExit.UserId with its value is padded to 8 characters with space.

Property

```
static public string TerminalId
```

Purpose

This read-only property provides the terminal ID string attribute.

Default Behavior

The default behavior is returning the value in the property SessionIdExit. TerminalId with its value is padded to 8 characters with space.

Property

```
static public string DialectName
```

Purpose

This read-only property provides the current dialect name for the load module.

Default Behavior

The default behavior is returning the value initialized in constructor. For more detail, please see the constructor section above.

Method

```
public static int GetDefaultYear (int  inYear, int currentYear)
```

Purpose

GetDefaultYear () is invoked when input editing occurs on a date or timestamp field, and the edit pattern specifies a 2 character year value. The 4-digit current year and the 2-digit input year are passed to GetDefaultYear (). By default, the current hundred year value is merely added to the 2 character year value and returned. This method allows the implementation of a customer-specified algorithm addressing Year-2000 concerns.

The default algorithm returns the current hundred years appended with two digit input year. The private method ImputeCenturies() enables an alternative algorithm, implementing a sliding range of hundred year values based on the input decade.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
inYear	Input	Integer containing the 2 or 4 digit year.
currentYear	Input	Integer containing the current year.

Return Code

Integer containing the 4 digit year.

Default Behavior

By default, this method invokes DefaultYearExit.GetDefaultYear(inYear, currentYear) with the given argument to this method and the return value is from the invoked method.

Method

```
private static string padAndTrim (string str, int length, char padChar)
```

Purpose

This method in used internally to trim and pad the given string with the specified arguments.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
str	Input	String that is passed in.
length	Input	Integer containing the length of the string.
padchar	Input	Char containing the pad character.

Return Code

The passed in string is padded and trimmed as specified.

Default Behavior

The passed in string is padded and trimmed as specified.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

Exits in <CAGen-root>\.net\exits\src\Common.cs file Used by CA Gen ASP.NET Web Clients and CA Gen .NET Servers

This section describes the exits that are located in the <CAGen-root>\.net\exits\src\Common.cs file and used by both the CA Gen ASP.NET Web Clients and CA Gen .NET Servers.

com.ca.gen.exits.common.CompareExit – C# Compare Exit

Source Code

Common.cs

Purpose

CompareExit is a runtime class used to compare various classes/types with each other.

Method

```
public static int CompareTo(decimal parm1, decimal parm2)
```

Purpose

Compares two decimals and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a decimal to be compared with
parm2	Input	a decimal to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to parm1.CompareTo() method.

Method

```
public static int CompareTo(char parm1, char parm2)
```

Purpose

Compares two chars and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a char to be compared with
parm2	Input	a char to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to parm1.CompareTo() method.

Method

```
public static int CompareTo(double parm1, double parm2)
```

Purpose

Compares two doubles and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a double to be compared with
parm2	Input	a double to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to parm1.CompareTo() method.

Method

```
public static int CompareTo(float parm1, float parm2)
```

Purpose

Compares two floats and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a float to be compared with
parm2	Input	a float to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to parm1.CompareTo() method.

Method

```
public static int CompareTo(int parm1, int parm2)
```

Purpose

Compares two ints and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a int to be compared with
parm2	Input	a int to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to parm1.CompareTo() method.

Method

```
public static int CompareTo(long parm1, long parm2)
```

Purpose

Compares two longs and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a long to be compared with
parm2	Input	a long to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to parm1.CompareTo() method.

Method

```
public static int CompareTo(object parm1, object parm2)
```

Purpose

Compares two objects and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

For comparison purposes, nulls are considered equal and a null compared to a non-null is always less than the non-null value.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	an object to be compared with
parm2	Input	an object to be compared to

Return Code

A negative, zero or a positive integer as this object is less than, equal to, or greater than the specified object.

Also this method may throw the InvalidCastException if the object does not implement a 'int CompareTo(Object)' interface like the Comparable interface.

Default Behavior

Depending on the type of parameters, this method delegate the comparison to CompareTo(parm1, parm2) methods defined in this class. If the type of the parameters does not match any of CompareTo(parm1, parm2) method, the method uses .NET Reflection mechanism to find the appropriate CompareTo method for the given types.

Method

```
public static int CompareTo(short parm1, short parm2)
```

Purpose

Compares two shorts and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a short to be compared with
parm2	Input	a short to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to parm1.CompareTo() method.

Method

```
public static int CompareTo(string parm1, string parm2)
```

Purpose

Compares two strings and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

The lengths do not have to be identical, they will still compare as long as characters in the extra length area of the larger string are spaces.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a string to be compared with
parm2	Input	a string to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

The default behavior of this method is as follows.

Check null for both parameters.

Trim the space from strings.

Delegate the comparison to parm1.CompareTo() method.

Method

```
public static int CompareTo(string parm1, string parm2 , int length)
```

Purpose

Compares two strings and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal. Only the characters up to the indicated length are compared. All characters after that point are ignored.

The lengths do not have to be identical, they will still compare as long as characters in the extra length area of the larger string are spaces.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a string to be compared with
parm2	Input	a string to be compared to
length	Input	an int to indicate the length of the strings to compare

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

The default behavior of this method is as follows.

Check null for both parameters.

Trim the given strings to the specified length.

Delegate the comparison to parm1.CompareTo() method.

Method

```
public static int CompareTo(DateTime parm1, DateTime parm2)
```

Purpose

Compares two DateTime instances and returns a negative if the first instance is less than the second; a positive if the first instance is greater than the second; 0 if the two instances are equal.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
parm1	Input	a DateTime to be compared with
parm2	Input	a DateTime to be compared to

Return Code

A negative, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Default Behavior

Delegate the comparison to parm1.CompareTo() method.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.common.LowerCaseExit – C# Lower Case Exit

Source Code

Common.cs

Purpose

LowerCaseExit is a runtime class used to lowercase the given string.

Method

```
public static string LowerCase(string inStr)
```

Purpose

LowerCase is a used to convert a given string to lowercase, returning a string as well.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
inStr	Input	a string to be lowercased

Return Code

A string representing the lowercased version of this object.

Default Behavior

By default, the string.toLower() function is used.

A null input parameter will result in a empty string being returned.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.common.UpperCaseExit – C# Upper Case Exit

Source Code

Common.cs

Purpose

UpperCaseExit is a runtime class used to lowercase the given string.

Method

```
public static string UpperCase(string inStr)
```

Purpose

UpperCase is a used to convert a given string to uppercase, returning a string as well.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
inStr	Input	a string to be uppercased

Return Code

A string representing the uppercased version of this object.

Default Behavior

By default, the string.toUpper() function is used.

A null input parameter will result in a empty string being returned.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.common.WebServiceMethodCallExit- C # CALL EXTERNAL User Exit

Source Code

Common.cs

Purpose

This user exit provides one method to modify the URL used to access a web service method at runtime and another method to modify the default truncation behavior used for return values during a web service call.

modifyURL Method-Modifies the URL

```
Public Static Final String modifyURL(String url)
```

Purpose

This method allows the URL used to access a web service method to be modified at runtime.

Arguments

The following table contains the arguments of the method:

Name	Output	Description
String	URL	The URL of the web service.

Return Code

The string representation of the URL of the web service.

Default Behavior

This method returns the URL of the web service that was added when the call external statement was created. The default behavior is to return the URL parameter unchanged.

ABRT_xcall_ws_gentype_truncate_exit —CALL EXTERNAL Data Truncation

```
long ABRT_xcall_ws_gentype_truncate_exit (void)
```

Purpose

By default, when the size of the web service response is greater than the matched Gen data type, the response is truncated. The truncation is governed by the table below.

Use this user exit to override the default behavior. The user exit sets the matched Gen data type flag so that the response is not truncated but an error is raised instead.

ABRT XCall WS Gentype Truncation Table:

The following table depicts the default behavior. The user exit sets the flag represented in the Overrides the Flag column. The default behavior overrides the flag.

XSD data type (from)	Gen data type (to)	Default truncation behavior	Overrides the flag
string	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
string	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB

XSD data type (from)	Gen data type (to)	Default truncation behavior	Overrides the flag
anyURI	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
QName	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
NOTATION	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
duration	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
base64Binary	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
base64Binary	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB
hexBinary	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
hexBinary	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB
float	Number (length <= 15, decimal points !=0, decimal precision = false)	Data will be truncated at decimal places.	GENTYPE_NUMERIC_DOUBLE
double	Number (length <= 15, decimal points !=0, decimal precision = false)	Data will be truncated at decimal places.	GENTYPE_NUMERIC_DOUBLE
decimal	Number (length <=18, decimal precision =true)	Data will be truncated at decimal places and other truncation results in Error.	GENTYPE_NUMERIC_PRECISION
short unsignedshort	Number (length <= 4)	Data will be truncated if it is > SHORT_MAX	GENTYPE_NUMERIC_SHORT

XSD data type (from)	Gen data type (to)	Default truncation behavior	Overrides the flag
integer long int nonPositiveInteger nonNegativeInteger negativeInteger positiveInteger unsignedLong unsignedInt	Number (length <= 9)	Data will be truncated if it is > LONG_MAX	GENTYPE_NUMERIC_LONG

Note: Data map pairs which are not mentioned in the table are not considered for truncation. The following XSD data types are not truncated:

- boolean
- datetime
- Time
- Date
- gYearMonth
- gYear
- gMonthDay
- gDay
- gMonth
- Bytes

For example, when the web service response of XSD datetime data type are mapped to Text, the response is not truncated.

Arguments

None.

Return Code

A long value that represents one or more flags which specify that the web service response for the given data types are not truncated but instead an error is raised instead.

Default Behavior

If the web service response size is greater than the Gen data type, the web service response is truncated. For more information about the default behavior, see the ABRT XCall WS Gentype Truncation Table in the Purpose topic.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

Related User Exits

None.

Exits in <CAGen-root>\.net\exits\src\msgobj Directory Used by CA Gen ASP.NET Web Clients and CA Gen .NET Servers

This section describes the exits that are located in the <CAGen-root>\.net\exits\src\msgobj directory and used by both the CA Gen ASP.NET Web Clients and CA Gen .NET Servers.

com.ca.gen.exits.msgobj.cfb.CFBDynamicMessageDecryptionExit – C# CFB Dynamic Message Decryption

Source Code

CFBDynamicMessageDecryptionExit.cs

Purpose

This class will be called after a CFB message has been received from a server.

Constructor

```
private CFBDynamicMessageDecryptionExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The CFBDynamicMessageDecryptionExit object.

Default Behavior

This constructor returns The CFBDynamicMessageDecryptionExit object.

Method

```
public static CFBDynamicMessageDecryptionExit GetInstance()
```

Purpose

This method will be invoked after a CFB message has been received from a server. This method obtains an instance of CFBDynamicMessageDecryptionExit class and initializes it with the private Init().

Arguments

None

Return Code

This method returns the initialized CFBDynamicMessageDecryptionExit object.

Default Behavior

The method uses a simple caching mechanism to obtain an instance of CFBDynamicMessageDecryptionExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

```
public void FreeInstance()
```

Purpose

This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

Method

```
public byte [] decryptData(byte [] data, int maxLength)
```

Purpose

This method is used to allow the user to decrypt the data portion of the message.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
Data	Input	The byte array to be decrypted.

Name	I/O	Description
maxLength	Input	Integer parameter indicates the largest byte array that can be returned.

Return Code

The return value is the byte array containing the decrypted data.

Default Behavior

By default the byte array passed in is unchanged and no real decryption occurs. The user should code their own algorithm to decrypt the data.

The byte array passed in can be modified in place and then returned, or if needed a local byte array can be allocated (larger or smaller if needed), populated and returned.

To indicate an error condition, throw a CSUException. That is, throw new CSUException("CFBDynamicMessageDecryptionExit:GetInstance","error message")).

Method

```
private void Init()
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is nothing.

Property

```
public byte DecryptionType
```

Purpose

This read-only byte property contains the value to specify whether decryption should be done. The valid values are:

CFBDynamicMessageDecryptionExit.DECRYPTION_NO

CFBDynamicMessageDecryptionExit.DECRYPTION_YES

Default Behavior

The default behavior is returning the value CFBDynamicMessageDecryptionExit.DECRYPTION_NO.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.msgobj.cfb.CFBDynamicMessageEncodingyExit – C# CFB Dynamic Message Encoding Exit

Source Code

CFBDynamicMessageEncodingExit.cs

Purpose

This class will be called from the CFBDynamicMessage.

Constructor

```
public CFBDynamicMessageEncodingExit()
```

Purpose

This is a default constructor.

Arguments

None

Return Code

This constructor returns The CFBDynamicMessageEncodingExit object.

Default Behavior

This constructor returns The CFBDynamicMessageEncodingExit object.

Method

```
public static int ServerCodePage(String tran)
```

Purpose

This method will be called to retrieve the message text encoding for the named host and transaction.

The runtimes will always call ServerCodePage to find the correct encoding for each transaction on each server before building the common format message to be sent.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
Tran	Input	Transaction name string associated with this request.

Return Code

This method returns the default encoding.

Default Behavior

The default behavior returns the default encoding. This behavior may be change to return any valid encoding number. The tran argument may be used to select different encodings for each transaction.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.msgobj.cfb.CFBDynamicMessageEncryptionExit – C# CFB Dynamic Message Encryption Exit

Source Code

CFBDynamicMessageEncryptionExit.cs

Purpose

This class will be called from the CFBDynamicMessage to encrypt message.

Constructor

```
private CFBDynamicMessageEncryptionExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The CFBDynamicMessageEncryptionExit object.

Default Behavior

This constructor returns The CFBDynamicMessageEncryptionExit object.

Method

```
public static CFBDynamicMessageEncryptionExit
        GetInstance(string newTranCode,
        string newNextLocation,
        string newUserid)
```

Purpose

This method obtains an instance of CFBDynamicMessageEncryptionExit class and initializes it with the private Init().

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newTranCode	Input	String containing the new tranCode
newNextLocation	Input	String containing the new nextLocation
newUserid	Input	String containing the new userID

Return Code

This method returns the initialized CFBDynamicMessageEncryptionExit object.

Default Behavior

The method uses a simple caching mechanism to obtain an instance of CFBDynamicMessageEncryptionExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

```
public void FreeInstance()
```

Purpose

This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

Method

```
public byte [] encryptData(byte [] data, int maxLength)
```

Purpose

This method is used to allow the user to encrypt the data portion of the message.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
Data	Input	The byte array to be encrypted.
maxLength	Input	Integer parameter indicates the largest byte array that can be returned.

Return Code

The return value is the byte array containing the encrypted data.

Default Behavior

By default the byte array passed in is unchanged and no real encryption occurs. The user should code their own algorithm to encrypt the data.

The byte array passed in can be modified in place and then returned, or if needed a local byte array can be allocated (larger or smaller if needed), populated and returned.

To indicate an error condition, throw a CSUException. That is, throw new CSUException("CFBDynamicMessageEncryptionExit:GetInstance", "error message")).

Method

```
private void Init(string newTranCode,
        string newNextLocation,
        string newUserid)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newTranCode	Input	String containing the new tranCode
newNextLocation	Input	String containing the new nextLocation
newUserid	Input	String containing the new userID

Return Code

None

Default Behavior

The default behavior of the method is assigning the specified argument to instance field.

Property

```
public byte EncryptionType
```

Purpose

This read-only byte property contains the value to specify whether encryption should be done. The valid values are:

CFBDynamicMessageEncryptionExit.ENCRYPTION_NO

CFBDynamicMessageEncryptionExit.ENCRYPTION_YES

Default Behavior

The default behavior is returning the value CFBDynamicMessageEncryptionExit.ENCRYPTION_NO.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.msgobj.cfb.CFBDynamicMessageSecurityExit – C# CFB Dynamic Message Security Exit

Source Code

CFBDynamicMessageSecurityExit.cs

Purpose

This class will be called from the CFBDynamicMessage.

Constructor

```
private CFBDynamicMessageSecurityExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The CFBDynamicMessageSecurityExit object.

Default Behavior

This constructor returns The CFBDynamicMessageSecurityExit object.

Method

```
public static CFBDynamicMessageSecurityExit
        GetInstance (string newUserid,
        string newPassword,
        string newTranCode,
        string newNextLocation)
```

Purpose

This method obtains an instance of CFBDynamicMessageSecurityExit class and initializes it with the private Init().

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newUserid	Input	String containing the new userID
newPassword	Input	String containing the new password
newTranCode	Input	String containing the new tranCode
newNextLocation	Input	String containing the new nextLocation

Return Code

This method returns the initialized CFBDynamicMessageSecurityExit object.

Default Behavior

The method uses a simple caching mechanism to obtain an instance of CFBDynamicMessageSecurityExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

```
public void FreeInstance()
```

Purpose

This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

Method

```
public byte [] getSecurityToken(int maxLength)
```

Purpose

This method is used to allow the user to pass back a security token to be passed in the enhanced security mode.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
maxLength	Input	Integer parameter indicates the largest byte array that can be returned.

Return Code

The return value is the byte array containing a security token.

Default Behavior

By default a 0 length array is returned.

To indicate an error condition, throw a CSUException. That is, throw new CSUException("CFBDynamicMessageSecurityExit:GetInstance","error message")).

Method

```
private void Init(string newUserid,
        string newPassword,
        string newTranCode,
        string newNextLocation)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
newUserid	Input	String containing the new userID
newPassword	Input	String containing the new password
newTranCode	Input	String containing the new tranCode
newNextLocation	Input	String containing the new nextLocation

Return Code

None

Default Behavior

The default behavior of the method is assigning the specified argument to instance field.

Property

```
public byte SecurityType
```

Purpose

This read-only byte property contains the value to specify what type of security should be used. The valid return values are:

CFBDynamicMessageSecurityExit.SECURITY_NO

CFBDynamicMessageSecurityExit.SECURITY_STANDARD

CFBDynamicMessageSecurityExit.SECURITY_ENHANCED

Default Behavior

The default behavior is returning the value:

CFBDynamicMessageSecurityExit.SECURITY_NO

Property

```
public bool useCMSecurity
```

Purpose

This read-only boolean property contains the value to specify whether the Client Manager/Comm Bridge to use the userID and password values for enhanced security validation or the standard Client Manager/Comm Bridge target server security configuration.

If the property contains true, the Client Manager/Comm Bridge uses the userID and password values for enhanced security validation.

If the property contains false, the Client Manager/Comm Bridge uses the standard Client Manager/Comm Bridge target server security configuration.

Default Behavior

The default behavior is returning the false.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

CA Gen .NET Servers

This section describes the exits that are located in the <CAGen-root>\.net\exits\src\scrt directory and used by the CA Gen .NET Servers.

com.ca.gen.exits.scrt.AuthorizationExit – C# Server Authorization Exit

Source Code

AuthorizationExit.cs

Purpose

This class provides the methods for the security interface. The methods in the class are called by the server manager. This module may be customized by the customer to perform transaction-level security.

Constructor

private AuthorizationExit()

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The AuthorizationExit object.

Default Behavior

This constructor returns The AuthorizationExit object.

Method

```
public static AuthorizationExit
        GetInstance(string loadModuleName,
        string procedureStepName)
```

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
loadModuleName	Input	string containing the loadModuleName
procedureStepName	Input	string containing the procedureStepName

Return Code

This method obtains an instance of AuthorizationExit class and initializes it with the private Init().

Default Behavior

The method uses a simple caching mechanism to obtain an instance of AuthorizationExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

```
public void FreeInstance()
```

Purpose

This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

Method

```
public void Authorize()
```

Purpose

This method will be invoked in server procedure to provide authorization.

Arguments

None

Return Code

To indicate an error condition, throw a GenException (i.e. throw new GenException("error message")).

To indicate an access failure, throw an UnauthorizedAccessException.

Default Behavior

The default behavior of the method is nothing.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.scrt.SecurityValidationExit – C# Server Security Validation Exit

Source Code

SecurityValidationExit.cs

Purpose

This class provides the methods to validate client security with the given Client Userid, Client Password, and SecurityObject parameters.

Constructor

```
private SecurityValidationExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The SecurityValidationExit object.

Default Behavior

This constructor returns The SecurityValidationExit object.

Method

```
public static SecurityValidationExit
        GetInstance (string clientUserId,
        string clientPassword,
        object securityObject)
```

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
clientUserId	Input	String containing the client user Id
clientPassword	Input	String containing the client password
securityObject	Input	Instance of security object

Return Code

This method obtains an instance of SecurityValidationExit class and initializes it with the private Init().

Default Behavior

The method uses a simple caching mechanism to obtain an instance of SecurityValidationExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

```
public void FreeInstance()
```

Purpose

This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

Method

```
public void Validate()
```

Purpose

This method will be invoked in server procedure to provide authorization.

Arguments

None

Return Code

To indicate an error condition, throw a GenException (i.e. throw new GenException("error message")).

To indicate an access failure, throw an UnauthorizedAccessException.

Default Behavior

The default behavior of the method is nothing.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.scrt.LocaleExit – C# Server Locale Exit

Source Code

LocaleExit.cs

Purpose

This exit class provides a set of methods that are called at startup of application to load customer-specific values for locale editing of data displayed in application. Each method provided by this exit is called with the default input, derived from the dialect specified during application design, and should return the appropriate localized value.

Method

```
public static char getLocalCurrencySymbol(char def)
```

Purpose

getLocalCurrencySymbol() supplies the currency symbol to the generated .NET application. The currency symbol is used when editing numeric fields, which includes currency symbol. For example, if the edit pattern is "@ZZZ.ZZZ,99", the getLocalCurrencySymbol() should specify "@" as the currency symbol.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
def	Input	Character containing the default currency symbol.

Return Code

Character containing the localized currency symbol.

Default Behavior

By default, Dollar sign '$' is returned.

Method

```
public static char getLocalThousandsSep(char def)
```

Purpose

getLocalThousandsSep () supplies the thousand separator to the generated .NET application. The thousand separator is used when editing numeric fields, which includes the thousand separator. For example, if the edit pattern is "@ZZZ.ZZZ,99", the getLocalThousandsSep () should specify "." as the thousand separator.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
def	Input	Character containing the default thousands separator.

Return Code

Character containing the localized thousands separator.

Default Behavior

By default, thousand separator passed in is returned.

Method

```
public static char getLocalDecimalSep(char def)
```

Purpose

getLocalDecimalSep () supplies the decimal point to the generated .NET application. The decimal point is used when editing numeric fields, which includes decimal point. For example, if the edit pattern is “@ZZZ.ZZZ,99”, the getLocalDecimalSep () should specify “,” as the decimal point.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
def	Input	Character containing the default decimal separator.

Return Code

Character containing the localized decimal point.

Default Behavior

By default, decimal point passed in is returned.

Method

```
public static char getLocalDateSep(char def)
```

Purpose

getLocalDateSep () supplies the date separator character to the generated .NET application. The date separator character is used only for date and time fields where the model does not specify the edit pattern. In these cases, the .NET runtime uses this information to build a default edit pattern using the information provided by the getLocalDateSep(). For example, if the getLocalDateSep() specifies the date separator as "-" (a dash) and the date order is yymmdd, then the default date edit pattern is yy-mm-dd.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
def	Input	Character containing the default date separator.

Return Code

Character containing the localized date separator.

Default Behavior

By default, date separator passed in is returned.

Method

```
public static char getLocalTimeSep(char def)
```

Purpose

getLocalTimeSep () supplies the time separator character to the generated .NET application. The time separator character is used only for date and time fields where the model does not specify the edit pattern. In these cases, the .NET runtime uses this information to build a default edit pattern using the information provided by the getLocalTimeSep (). For example, if the getLocalTimeSep () specifies the date separator as ":" (a colon) then the default date edit pattern is yy:mm:dd.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
def	Input	Character containing the default time separator.

Return Code

Character containing the localized time separator.

Default Behavior

By default, time separator passed in is returned.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.scrt.RetryLimitExit – C# Server Retry Limit Exit

Source Code

RetryLimitExit.cs

Purpose

This exit class provides configuration of the number of times a procedure step can be retried when the application has requested a "retry transaction", or when a deadlock condition on the database has been detected.

Property

```
public static int UltimateRetryLimit
```

Purpose

This property is a Integer containing absolute upper limit to the number of times a procedure step can be retried. This exit provides a safeguard in case the system attribute "transaction retry limit" is set to an excessive value by an action diagram. This exit defines the upper bound to the retry limit value which can never be exceeded.

Default Value

By default, "99" is returned.

Property

```
public static int DefaultRetryLimit
```

Purpose

This property is Integer containing default retry limit to the number of times a Procedure step can be retried in the event that the system attribute "Transaction retry limit" is not set by an action diagram.

Default Value

By default, “10” is returned.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with ‘Run as administrator’.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.scrt.SrvrErrorExit – C# Server Error Exit

Source Code

SrvrErrorExit.cs

Purpose

This exit class provides methods that are invoked when an error is detected during the processing of a server to server flow.

Method

```
public static int ServerError(string toPstep,
        string fromPstep,
        string errorList,
        int failureType))
```

Purpose

ServerError() is invoked when an error is detected during the processing of a server to server flow. The parameter failureType describes the origin of this error.

The exit can influence the default runtime error behavior.

When the NOTPROPAGATE_ERR is returned, the calling procedure step continues the execution ignoring the fact that error occurs in the called procedure step.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
toPstep	Input	String contains the target procedure step.
fromPstep	Input	String contains the calling procedure step.
errorList	Input	String contains error message.
failureType	Input	An integer value describing the source of the failure. It's value can be one of the following: com.ca.gen.scrt. SrvrErrorExit.CFBUILD or '0': An error in the construction or parsing of a client/server flow message or response. com.ca.gen.scrt. SrvrErrorExit.XFAL or '1': An error during the server procedures action block execution. com.ca.gen.scrt. SrvrErrorExit.XERR or '2': A communications error occurring somewhere between construction of a message or response, and the deciphering of that message by the partner in this flow.

Return Code

The following table gives a brief description of each of the return codes.

Return Code	Description
com.ca.gen.scrt.SrvrErrorExit.PROPAGATE_ERR	This return value causes the calling procedure step stops the execution and propagates error. The integer value is '0'
com.ca.gen.scrt.SrvrErrorExit.NOTPROPAGATE_ERR	This return value causes the calling procedure step continues the execution and ignoring the fact that error occurs in the called procedure step. The integer value is '1'

Default Behavior

The default implementation always returns the value com.ca.gen.scrt.SrvrErrorExit.NOTPROPAGATE_ERR causes the calling procedure step continues the execution and ignoring the fact that error occurs in the called procedure step.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.scrt.UserExit – C# Server User Exit

Source Code

UserExit.cs

Purpose

This class provides the methods to use UserExit object.

Constructor

```
private UserExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns the UserExit object.

Default Behavior

This constructor returns the UserExit object.

Method

```
public static UserExit
        GetInstance (object userObject)
```

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
userObject	Input	Instance of user object

Return Code

This method obtains an instance of UserExit class and initializes it with the private Init().

Default Behavior

The method uses a simple caching mechanism to obtain an instance of UserExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

```
public void FreeInstance()
```

Purpose

This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

Method

```
public void UseRuntimeObject()
```

Purpose

This method will be invoked in server procedure to use user object.

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is nothing.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

C# Server Middleware User Exits

com.ca.gen.exits.coopflow.net.NETDynamicCoopFlowExit–C# NET Dynamic CoopFlow Exit

Source Code

NETDynamicCoopFlowExit.cs

Purpose

The methods in the class will be called prior to performing a connection from the NETDynamicCoopFlow. The class will be instantiated with various data and methods will be called to override that data.

Constructor

private NETDynamicCoopFlowExit()

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The NETDynamicCoopFlowExit object.

Default Behavior

This constructor returns The NETDynamicCoopFlowExit object.

Method

```
public static NETDynamicCoopFlowExit GetInstance (
        string programID,
        string tranCode,
        string procedureName,
        string procedureSourceName,
        string modelName,
        string modelShortName,
        string netApplicationName,
        string netNamespace,
        string netAssemblyVersion,
        string nextLocation,
        string hostName,
        int port,
        char protocolCode)
```

Purpose

This method is invoked at the beginning of performing a connection from the NETDynamicCoopFlow. This method obtains an instance of NETDynamicCoopFlowExit class and initializes it with the specified parameter.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
programID	Input	string contains Program ID
tranCode	Input	string contains Transaction Code
procedureName	Input	string contains procedure name
procedureSourceName	Input	string contains source file name for procedure
modelName	Input	string contains Model name
modelShortName	Input	string contains Model short same
netApplicationName	Input	string contains Application name

Name	I/O	Description
netNameSpace	Input	string contains application name space
netAssemblyVersion	Input	string contains application assembly version
nextLocation	Input	string contains next Location
hostname	Input	string contains Host Name
port	Input	integer contains Port number
protocolCode	Input	Character contains Protocol Code. The possible values are: B: HTTP with binary encoding. (Default) S: Soap T: TCP/IP (will not be supported as of now)

Return Code

This method returns the initialized NETDynamicCoopFlowExit object.

Default Behavior

The method uses a simple caching mechanism to obtain an instance of NETDynamicCoopFlowExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

```
public void FreeInstance()
```

Purpose

At the end of performing a connection from the NETDynamicCoopFlow, This method will be invoked to de-allocate the object obtained with GetInstance ().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

Method

```
public bool ProcessException(int attempts,
    Exception e)
```

Purpose

This method will be invoked whenever the Coopflow fails to either instantiate the remote object or the server call fails.

Use this exit to indicate whether to retry the operation or to throw an exception.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
attempts	Input	Integer containing the number of attempts
e	Input	Exception

Return Code

The method returns true to retry the flow and false to go ahead and process/throw the exception.

Default Behavior

By default, the method returns false in any case. Also if tracing is enabled, the given exception object is recorded in the trace.

Method

```
private void Init(string programID,
        string tranCode,
        string procedureName,
        string procedureSourceName,
        string modelName,
        string modelShortName,
        string netApplicationName,
        string netNamespace,
        string netAssemblyVersion,
        string nextLocation,
        string hostName,
        int port,
        char protocolCode)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
programID	Input	string contains Program ID
tranCode	Input	string contains Transaction Code
procedureName	Input	string contains procedure name
procedureSourceName	Input	string contains source file name for procedure
modelName	Input	string contains Model name
modelShortName	Input	string contains Model short same
netApplicationName	Input	string contains Application name
netNameSpace	Input	string contains application name space
netAssemblyVersion	Input	string contains application assembly version
nextLocation	Input	string contains next Location
hostname	Input	string contains Host Name
port	Input	integer contains Port number
protocolCode	Input	Character contains Protocol Code. The possible values are: B: HTTP with binary encoding. (Default) S: Soap T: TCP/IP (will not be supported as of now)

Return Code

None

Default Behavior

The default behavior of the method is simply duplicate the specified parameter to the corresponding instance property.

Property

```
public virtual string hostName
```

Purpose

This read-only string property contains the Host Name.

Default Behavior

The default behavior is returning the value initialized in private Init() method.

Property

```
public int port
```

Purpose

This read-only integer property contains the port number.

Default Behavior

The default behavior is returning the value initialized in private Init() method.

Property

```
public char protocolCode
```

Purpose

This read-only character property contains the code to specify protocol to be used. The possible values are:

B: HTTP with binary encoding. (Default)

S: Soap

T: TCP/IP (will not be supported as of now)

Default Behavior

The default behavior is returning the value initialized in private Init() method.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

Exits in <CAGen-root>\.net\exits\src\coopflow directory Used by CA Gen ASP.NET Web Clients and CA Gen .NET Servers

This section describes the exits that are located in the <CAGen-root>\.net\exits\src\coopflow directory and used by both the CA Gen ASP.NET Web Clients and CA Gen .NET Servers.

com.ca.gen.exits.coopflow.complus.COMPLUSDynamicCoopFlowExit–C# COM PLUS Dynamic Coop Flow Exit

Source Code

COMPLUSDynamicCoopFlowExit.cs

Purpose

The methods in the class will be called prior to performing a COMPLUS connection from the COMPLUSDynamicCoopFlow. The class will be instantiated with various data and methods will be called to override that data.

Constructor

```
private COMPLUSDynamicCoopFlowExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The COMPLUSDynamicCoopFlowExit object.

Default Behavior

This constructor returns The COMPLUSDynamicCoopFlowExit object.

Method

```
public static COMPLUSDynamicCoopFlowExit GetInstance (
                    string programID,
                    string tranCode,
                    string procedureName,
                    string procedureSourceName,
                    string modelName,
                    string modelShortName,
                    string netApplicationName,
                    string netNamespace,
                    string netAssemblyVersion,
                    string nextLocation)
```

Purpose

This method is invoked at the beginning of performing a COMPLUS connection from the COMPLUSDynamicCoopFlow. This method obtains an instance of COMPLUSDynamicCoopFlowExit class and initializes it with the specified parameter.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
programID	Input	string contains Program ID
tranCode	Input	string contains Transaction Code
procedureName	Input	string contains procedure name

Name	I/O	Description
procedureSourceName	Input	string contains source file name for procedure
modelName	Input	string contains Model name
modelShortName	Input	string contains Model short same
netApplicationName	Input	string contains Application name
netNameSpace	Input	string contains application name space
netAssemblyVersion	Input	string contains application assembly version
nextLocation	Input	string next Location

Return Code

This method returns the initialized COMPLUSDynamicCoopFlowExit object.

Default Behavior

The method uses a simple caching mechanism to obtain a instance of COMPLUSDynamicCoopFlowExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

```
public void FreeInstance()
```

Purpose

At the end of performing a COMPLUS connection from the COMPLUSDynamicCoopFlow, This method will be invoked to de-allocate the object obtained with GetInstance ().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance () invocation.

Method

```
public bool ProcessException(int attempts,
    Exception e)
```

Purpose

This method will be invoked whenever the COMPLUIS Coopflow fails to either instantiate the remote object or the server call fails.

Use this exit to indicate whether to retry the operation or to throw an exception.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
attempts	Input	Integer containing the number of attempts
e	Input	Exception

Return Code

The method returns true to retry the flow and false to go ahead and process/throw the exception.

Default Behavior

By default, the method returns false in any case. Also if tracing is enabled, the given exception object is recorded in the trace.

Method

```
private void Init(string programID,
        string tranCode,
        string procedureName,
        string procedureSourceName,
        string modelName,
        string modelShortName,
        string netApplicationName,
        string netNamespace,
        string netAssemblyVersion,
        string nextLocation)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
programID	Input	string contains Program ID
tranCode	Input	string contains Transaction Code
procedureName	Input	string contains procedure name
procedureSourceName	Input	string contains source file name for procedure
modelName	Input	string contains Model name
modelShortName	Input	string contains Model short same
netApplicationName	Input	string contains Application name
netNameSpace	Input	string contains application name space
netAssemblyVersion	Input	string contains application assembly version
nextLocation	Input	string next Location

Return Code

None

Default Behavior

The default behavior of the method is simply duplicate the specified parameter to the corresponding instance property.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.coopflow.complus.COMPLUSDynamicCoopFlowSecurityExit–C# COMPLUS Dynamic Coop Flow Security Exit

Source Code

COMPLUSDynamicCoopFlowSecurityExit.cs

Purpose

The methods in the class will be called in the COMPLUSDynamicCoopFlow class to provide security object when invoking server procedures.

Constructor

```
private COMPLUSDynamicCoopFlowSecurityExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The COMPLUSDynamicCoopFlowSecurityExit object.

Default Behavior

This constructor returns The COMPLUSDynamicCoopFlowSecurityExit object.

Method

```
public static COMPLUSDynamicCoopFlowSecurityExit GetInstance (string tranCode,
string nextLocation)
```

Purpose

This method is invoked at the beginning of performing a COMPLUS connection from the COMPLUSDynamicCoopFlow. This method obtains an instance of COMPLUSDynamicCoopFlowSecurityExit class and initializes it with the specified parameter.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
tranCode	Input	string contains Transaction Code
nextLocation	Input	string next Location

Return Code

This method returns the initialized COMPLUSDynamicCoopFlowSecurityExit object.

Default Behavior

The method uses a simple caching mechanism to obtain a instance of COMPLUSDynamicCoopFlowSecurityExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

```
public void FreeInstance()
```

Purpose

At the end of performing a COMPLUS connection from the COMPLUSDynamicCoopFlow, This method will be invoked to de-allocate the object obtained with GetInstance ().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance () invocation.

Method

```
public Object getSecurityObject()
```

Purpose

This method instantiates and returns an object that is passed to the server as a security object.

Arguments

None

Return Code

The method returns the object that will be passed to server to as a security object.

Default Behavior

By default, the method returns newly instantiated Object class instance.

Method

```
private void Init(string tranCode, string nextLocation)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
tranCode	Input	string contains Transaction Code
nextLocation	Input	string next Location

Return Code

None

Default Behavior

The default behavior of the method is nothing.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.coopflow.mqs.MQSDynamicCoopFlowExit–C# MQSeries Dynamic Coop Flow Exit

Source Code

MQSDynamicCoopFlowExit.cs

Purpose

This class will be called prior to performing an MQSeries connection from the MQSDynamicCoopFlow. The class will be instantiated with various data and methods will be called to override that data. In addition, the MQSeries C# client may need to have some MQEnvironment modifications. These customizations are performed in this exit.

Constructor

public MQSDynamicCoopFlowExit

Purpose

The default constructor for this class is public and is invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns the MQSDynamicCoopFlowExit object.

Default Behavior

This constructor returns the MQSDynamicCoopFlowExit object.

Method

```
public static MQSDynamicCoopFlowExit GetInstance(string qMgrName,
                                                 string remoteQMgrName,
                                                 string putQName,
                                                 string replyModelQName,
                                                 string dynamicQName,
                                                 string nextLocation,
                                                 string programID,
                                                 string tranCode,
                                                 string procedureName,
                                                 string
        procedureSourceName,
                                                 string modelName,
                                                 string modelShortName,
                                                 string
        netApplicationName,
                                                 string netNamespace,
                                                 string
        netAssemblyVersion,
                                                 int reportOptions)
```

Purpose

This method will be invooked to retrieve an instance of the exit class. By default, a new instance is created for each request if one does not already exist in the free array.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
qMgrName	Input	string contains queue manager name
remoteQMgrName	Input	string contains remote queue manager name
putQName	Input	string contains put queue name
replyModelQName	Input	string contains reply queue name
dynamicQName	Input	string contains dynamic queue name

Name	I/O	Description
nextLocation	Input	string next Location
programID	Input	string contains Program ID
tranCode	Input	string contains Transaction Code
procedureName	Input	string contains procedure name
modelName	Input	string contains Model name
modelShortName	Input	string contains Model short same
netApplicationName	Input	string contains Application name
netNamespace	Input	string contains application name space
netAssemblyVersion	Input	string contains application assembly version
reportOptions	Input	integer contains mask of MQSeries options

Return Code

This method returns the initialized MQSDynamicCoopFlowExit object.

Default Behavior

The method uses a simple caching mechanism to obtain a instance of MQSDynamicCoopFlowExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

public void FreeInstance()

Purpose

At the end of performing a MQSeries connection from the MQSDynamicCoopFlowExit, This method will be invoked to de-allocate the object obtained with GetInstance ().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance () invocation.

Method

```
public bool ProcessException(int attempts,
    Exception e)
```

Purpose

This method will be invoked whenever the COMPLUIS Coopflow fails to either instantiate the remote object or the server call fails.

Use this exit to indicate whether to retry the operation or to throw an exception.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
attempts	Input	Integer containing the number of attempts
e	Input	Exception

Return Code

The method returns true to retry the flow and false to go ahead and process/throw the exception.

Default Behavior

By default, the method returns false in any case. Also if tracing is enabled, the given exception object is recorded in the trace.

Property

QMgrName

Purpose

This property is used to retrieve the QMgrName to be used for the MQSeries communications.

Default Behavior

The default value is set by the constructor with the original value the runtime contained.

Property

RemoteQMgrName

Purpose

This property is used to retrieve the PutQName to be used for the MQSeries communications.

Default Behavior

The default value is set by the constructor with the original value the runtime contained.

Property

PutQName

Purpose

This property is used to retrieve the PutQName to be used for the MQSeries communications.

Default Behavior

The default value is set by the constructor with the original value the runtime contained.

Property

ReplyModelQName

Purpose

This property is used to retrieve the ReplyModelQName to be used for the MQSeries communications.

Default Behavior

The default value is set by the constructor with the original value the runtime contained.

Property

ReplyTimeout

Purpose

This property is used to retrieve the ReplyTimeout to be used for the MQSeries communications.

Default Behavior

The default value is MQC.MQWI_UNLIMITED (-1).

Property

ClosePutQ

Purpose

This property is used to determine if put queues must be closed after each cooperative flow.

Default Behavior

The default value is false.

Property

CloseGetQ

Purpose

This property is used to determine if get queues must be closed after each cooperative flow.

Default Behavior

The default value is false.

Property

DynamicQName

Purpose

This property is used to retrieve the dynamic queue that is created for the MQSeries communications.

Default Behavior

The default value supplied by the runtime is "USER_ID.THEAD_HASHCODE.PROCESS_ID.*" This means the MQSeries subsystem will generate a queue name based on this prefix followed by a series of uniquely generated numbers.

Note: Four valid characters must be provided prior to the "*" to create a dynamic queue name. For example: "tmpq*" succeeds but "tmp*" does not. Contact IBM MQSeries support for further assistance if this proves to be a problem. IBM APAR #IC46539 has been opened to address this problem.

Property

ReportOptions

Purpose

This property is used to let the user override the report options that have been set by the Gen runtimes.

Default Behavior

The default is what is passed as input and can be set to any value that is valid in the environment.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.coopflow.net.NETDynamicCoopFlowSecurityExit–C# NET Dynamic Coop Flow Security Exit

Source Code

NETDynamicCoopFlowSecurityExit.cs

Purpose

The methods in the class will be called in the NETDynamicCoopFlow class to provide security object when invoking server procedures.

Constructor

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The NETDynamicCoopFlowSecurityExit object.

Default Behavior

This constructor returns The NETDynamicCoopFlowSecurityExit object.

Method

```
public static NETDynamicCoopFlowSecurityExit GetInstance(string tranCode,
        string nextLocation,
        string hostName,
        int port)
```

Purpose

This method is invoked at the beginning of performing a connection from the NETDynamicCoopFlow. This method obtains an instance of NETDynamicCoopFlowSecurityExit class and initializes it with the specified parameter.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
tranCode	Input	string contains Transaction Code
nextLocation	Input	string contains next Location
hostname	Input	string contains Host Name
port	Input	integer contains Port number

Return Code

This method returns the initialized NETDynamicCoopFlowSecurityExit object.

Default Behavior

The method uses a simple caching mechanism to obtain a instance of NETDynamicCoopFlowSecurityExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

```
public void FreeInstance()
```

Purpose

At the end of performing a COMPLUS connection from the NETDynamicCoopFlow, This method will be invoked to de-allocate the object obtained with GetInstance ().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

Method

```
public Object getSecurityObject()
```

Purpose

This method instantiates and returns an object that is passed to the server as a security object.

Arguments

None

Return Code

The method returns the object that will be passed to server to as a security object.

Default Behavior

By default, the method returns newly instantiated Object class instance.

Method

```
private void Init(string tranCode,
        string nextLocation,
        string hostName,
        int port)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
tranCode	Input	string contains Transaction Code
nextLocation	Input	string contains next Location
hostname	Input	string contains Host Name
port	Input	integer contains Port number

Return Code

None

Default Behavior

The default behavior of the method is nothing.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.coopflow.tcpip.TCPIPDynamicCoopFlowExit–C# TCPIP Dynamic Coop Flow Exit

Source Code

TCPIPDynamicCoopFlowExit.cs

Purpose

The methods in the class will be called prior to performing a TCPIP connection from the TCPIPDynamicCoopFlow. The class will be instantiated with various data and methods will be called to override that data.

Constructor

```
private TCPIPDynamicCoopFlowExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The TCPIPDynamicCoopFlowExit object.

Default Behavior

This constructor returns The TCPIPDynamicCoopFlowExit object.

Method

```
public static TCPIPDynamicCoopFlowExit GetInstance (
        string programID,
        string tranCode,
        string procedureName,
        string procedureSourceName,
        string modelName,
        string modelShortName,
        string netApplicationName,
        string netNamespace,
        string netAssemblyVersion,
        string nextLocation,
        string hostName,
        int port,
        bool clientPersistence)
```

Purpose

This method is invoked at the beginning of performing a TCPIP connection from the TCPIPDynamicCoopFlow. This method obtains an instance of TCPIPDynamicCoopFlowExit class and initializes it with the specified parameter.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
programID	Input	string contains Program ID
tranCode	Input	string contains Transaction Code
procedureName	Input	string contains procedure name
procedureSourceName	Input	string contains source file name for procedure

Name	I/O	Description
modelName	Input	string contains Model name
modelShortName	Input	string contains Model short same
netApplicationName	Input	string contains Application name
netNameSpace	Input	string contains application name space
netAssemblyVersion	Input	string contains application assembly version
nextLocation	Input	string contains next Location
hostname	Input	string contains Host Name
port	Input	integer contains Port number
clientPersistence	Input	bool contains clientPersistence

Return Code

This method returns the initialized TCPIPDynamicCoopFlowExit object.

Default Behavior

The method uses a simple caching mechanism to obtain a instance of TCPIPDynamicCoopFlowExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

Method

```
public void FreeInstance()
```

Purpose

At the end of performing a TCPIP connection from the TCPIPDynamicCoopFlow, This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance () invocation.

Method

```
public bool ProcessException(int attempts,
        GenException e)
```

Purpose

This method will be invoked whenever the TCPIP Coopflow fails to either instantiate the remote object or the server call fails.

Use this exit to indicate whether to retry the operation or to throw an exception.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
attempts	Input	Integer containing the number of attempts
e	Input	GenException

Return Code

The method returns true to retry the flow and false to go ahead and process/throw the exception.

Default Behavior

By default, the method returns false in any case. Also if tracing is enabled, the given exception object is recorded in the trace.

Method

```
private void Init(string programID,
        string tranCode,
        string procedureName,
        string procedureSourceName,
        string modelName,
        string modelShortName,
        string netApplicationName,
        string netNamespace,
        string netAssemblyVersion,
        string nextLocation,
        string hostName,
        int port,
        bool clientPersistence)
```

Purpose

This private method is invoked internally from the GetInstance () to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
programID	Input	string contains Program ID
tranCode	Input	string contains Transaction Code
procedureName	Input	string contains procedure name
procedureSourceName	Input	string contains source file name for procedure
modelName	Input	string contains Model name
modelShortName	Input	string contains Model short same
netApplicationName	Input	string contains Application name
netNameSpace	Input	string contains application name space
netAssemblyVersion	Input	string contains application assembly version
nextLocation	Input	string contains next Location
hostname	Input	string contains Host Name
port	Input	integer contains Port number
clientPersistence	Input	bool contains clientPersistence

Return Code

None

Default Behavior

The default behavior of the method is simply duplicate the specified parameter to the corresponding instance property.

Property

```
public string hostName
```

Purpose

This read-only string property contains the Host Name.

Default Behavior

The default behavior is returning the value initialized in private Init() method.

Property

```
public int port
```

Purpose

This read-only integer property contains the port number.

Default Behavior

The default behavior is returning the value initialized in private Init() method.

Property

```
public bool clientPersistence
```

Purpose

This read-only boolean property contains the code to specify protocol to be used. The value is used to specify the desired persistence of the TCP/IP connection. If a value is true, the connection will be cached for subsequent reuse. The socket will not be closed at the completion of the server response. If a value is false, the connection will not be cached. The socket will be closed and the connection object will be destroyed. A subsequent flow to the same host/port combination will require a new connection object and a new socket connection.

Default Behavior

The default behavior is returning the value initialized in private Init() method.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with ‘Run as administrator’.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.

3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.

4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

com.ca.gen.exits.coopflow.ws.WSDynamicCoopFlowExit – C# WS Dynamic Coop Flow Exit

Source Code

WSDynamicCoopFlowExit.cs

Purpose

This exit is only provided for future use and is not currently used.

The methods in the class will be called from the WSDynamicCoopFlow. The class will be instantiated with various data and methods will be called to override that data.

Constructor

```
private WSDynamicCoopFlowExit()
```

Purpose

The default constructor for this class is private and is only invoked from the GetInstance() method to provide the default caching mechanism.

Arguments

None

Return Code

This constructor returns The WSDynamicCoopFlowExit object.

Default Behavior

This constructor returns The WSDynamicCoopFlowExit object.

GetInstance Method

```
public static WSDynamicCoopFlowExit GetInstance (
        string programID,
        string tranCode,
        string procedureName,
        string procedureSourceName,
        string modelName,
        string modelShortName,
        string netApplicationName,
        string netNamespace,
        string netAssemblyVersion,
        string nextLocation,
        string baseURL,
        string contextType)
```

Purpose

This method is invoked at the beginning of performing a WS connection from the WSDynamicCoopFlow. This method obtains an instance of WSDynamicCoopFlowExit class and initializes it with the specified parameter.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
programID	Input	string contains Program ID
tranCode	Input	string contains Transaction Code
procedureName	Input	string contains procedure name
procedureSourceName	Input	string contains source file name for procedure
modelName	Input	string contains Model name
modelShortName	Input	string contains Model short name
netApplicationName	Input	string contains Application name
netNameSpace	Input	string contains application name space
netAssemblyVersion	Input	string contains application assembly version
nextLocation	Input	string contains next Location
baseURL	Input	string contains base URL
contextType	Input	string contains contextType

Return Code

This method returns the initialized WSDynamicCoopFlowExit object.

Default Behavior

The method uses a simple caching mechanism to obtain an instance of WSDynamicCoopFlowExit class and initializes it with the private method Init() in the class. Then the method returns the initialized object.

FreeInstance Method

```
public void FreeInstance()
```

Purpose

At the end of performing a WS connection from the WSDynamicCoopFlow, This method will be invoked to de-allocate the object obtained with GetInstance().

Arguments

None

Return Code

None

Default Behavior

The default behavior of the method is simply returning the current instance into free instance array to be used for next GetInstance() invocation.

ProcessException Method

```
public bool ProcessException(int attempts,
        Exception e)
```

Purpose

This method will be invoked whenever the WS Coopflow fails to either instantiate the remote object or the server call fails.

Use this exit to indicate whether to retry the operation or to throw an exception.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
attempts	Input	Integer containing the number of attempts
e	Input	GenException

Return Code

The method returns true to retry the flow and false to go ahead and process/throw the exception.

Default Behavior

By default, the method returns false in any case. Also if tracing is enabled, the given exception object is recorded in the trace.

Init Method

```
private void Init(string programID,
        string tranCode,
        string procedureName,
        string procedureSourceName,
        string modelName,
        string modelShortName,
        string netApplicationName,
        string netNamespace,
        string netAssemblyVersion,
        string nextLocation,
        string baseURL,
        string contextType)
```

Purpose

This private method is invoked internally from the GetInstance() to initialize the current instance.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
programID	Input	string contains Program ID

Name	I/O	Description
tranCode	Input	string contains Transaction Code
procedureName	Input	string contains procedure name
procedureSourceName	Input	string contains source file name for procedure
modelName	Input	string contains Model name
modelShortName	Input	string contains Model short same
netApplicationName	Input	string contains Application name
netNameSpace	Input	string contains application name space
netAssemblyVersion	Input	string contains application assembly version
nextLocation	Input	string contains next Location
baseURL	Input	string contains base URL
contextType	Input	string contains contextType

Return Code

None

Default Behavior

The default behavior of the method simply duplicates the specified parameter to the corresponding instance property.

BaseURL Property

```
public string BaseURL
```

Purpose

This read-only string property contains the base URL.

Default Behavior

The default behavior is returning the value initialized in private Init() method.

ContextType Property

```
public string ContextType
```

Purpose

This read-only string property contains the contextType.

Default Behavior

The default behavior is returning the value initialized in private Init() method.

Rebuilding the Exit

This exit is built as part of a common user exit .NET assembly called CA.Gen.exits.dll. Prerequisites for building the assembly are Microsoft C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch a Command Prompt window. This Command Prompt should be opened with 'Run as administrator'.
2. Change your current directory to that containing the makeexits.bat file. Typically, this will be in the CA Gen installed area, .net\exits subdirectory.
3. To build the exit, run makeexits.bat.

   Note: See the makeexits.bat file for execution instructions.
4. Redeploy the resulting CA.Gen.exits.dll for use with the appropriate Applications.

Chapter 9: Browser User Exits

Customize userOnLoad in ASP.NET Mode

This is the only user exit that executes on the browser. It is invoked when the browser executes the onLoad handler of the page. That is, userOnLoad is executed after all other processing on the page has occurred. To add user-specified logic, an HTMLControl must be added to the page where the user exit is to be used. This is done through the Navigation Diagram in ASP.NET mode.

Click Add, HTML Control in the Toolset main menu. The HTML Control Properties dialog opens where you can insert the following JavaScript:

```
<script>
function userOnLoad()
{
//Insert your code here ***
}
</script>
```

The preceding code executes the userOnLoad() method for every request (every time the page with the exit loads). To execute the user exit only the first time the page loads, modify the code as follows (subsequent requests that result in the same page being returned will not execute the exit):

```
<script>
function userOnLoad()
{
if (isNewWindow == true)
{
//Insert your code here ***
}
}
</script>
```

Customize userOnLoad in HTML Mode for Web View

This is the only user exit that executes on the browser. It is invoked when the Window or Dialog Box has completed the loading and initialization process in the browser. That is, userOnLoad is executed after all other processing on the page has occurred. To add user-specified logic, an HTMLControl must be added to the page where the user exit is to be used. This is done through the Navigation Diagram in HTML mode.

Click Add, HTML Control in the Toolset main menu. The HTML Control Properties dialog opens where you can insert the following JavaScript:

```
<script>
function userOnLoad(theWindow) {
//Insert your code here ***
}
</script>
```

theWindow

Specifies the HTML DIV element that contains the contents of the window or dialog box that is rendered to the browser.

Chapter 10: Action Block Runtime User Exit

Windows Action Block Runtime User Exits

The following table summarizes the functions available through the action block user exits for C based applications:

ABRT: Language: C		
User Exit Name	**Source Code**	**Description**
ABRT_xcall_ws_url_exit	abrtexit.c	CALL EXTERNAL Web Service URL Exit
ABRT_xcall_ws_gentype_truncate_exit	abrtexit.c	CALL EXTERNAL Data Truncation Exit

ABRT user exits are rebuilt into the DLL ABEXxxN.DLL using the makefile abrtexit.nt in %GENxx%Gen\VSabc for Visual Studio 32-bit or %GENxx%Gen\VSabc\amd64 for Visual Studio 64-bit.

Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

ABRT_xcall_ws_url_exit —CALL EXTERNAL Web Service URL Exit

```
void ABRT_xcall_ws_url_exit (char *url,
size_t urlMaxLen)
```

Source Code

ABRTEXIT.C

Purpose

The action block runtime invokes this user exit to let a user influence the web service URL associated with a CALL EXTERNAL web service action block statement.

The inputs for this user exit are the URL and the maximum size variables. The URL input variable is modified as necessary. However, the length of the modified URL must not exceed the specified urlMaxLen variable.

For more information about the input and output fields of this exit routine, see Arguments.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*url	Input/Output	A pointer to a character array that contains the value of the web service URL. This user exit can set this value by modifying the data area pointed to by this argument.
urlMaxLen	Input	A size_t field that contains the maximum length allowed for the associated URL parameter.

Return Code

None

Default Behavior

The CALL EXTERNAL Web Service URL user exit, as delivered with CA Gen, will return without modifying the URL value.

Building on Windows

The CALL EXTERNAL Web Service URL exit is built as part of the dynamic link library ABEXxxN.DLL.

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*. N indicates platform.

Install the Microsoft Visual C++ compiler on your system as a prerequisite for building the DLL.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile ABRTEXIT.NT. The path is %GENxx%Gen\VSabc for Visual Studio 32-bit or %GENxx%Gen\VSabc\amd64 for Visual Studio 64-bit.

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set the Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F ABRTEXIT.NT.

Related User Exits

None.

ABRT_xcall_ws_gentype_truncate_exit —CALL EXTERNAL Data Truncation

```
long ABRT_xcall_ws_gentype_truncate_exit (void)
```

Source

ABRTEXIT.C

Purpose

By default, when the size of the web service response is greater than the matched Gen data type, the response is truncated. The truncation is governed by the table below.

Use this user exit to override the default behavior. The user exit sets the matched Gen data type flag so that the response is not truncated but an error is raised instead.

ABRT XCall WS Gentype Truncation Table:

The following table depicts the default behavior. The user exit sets the flag represented in the Overrides the Flag column. The default behavior overrides the flag.

XSD data type (from)	Gen data type (to)	Default truncation behavior	Overrides the flag
string	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
string	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB
anyURI	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
QName	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
NOTATION	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
duration	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
base64Binary	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
base64Binary	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB
hexBinary	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
hexBinary	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB

XSD data type (from)	Gen data type (to)	Default truncation behavior	Overrides the flag
float	Number (length <= 15, decimal points !=0, decimal precision = false)	Data will be truncated at decimal places.	GENTYPE_NUMERIC_DOUBLE
double	Number (length <= 15, decimal points !=0, decimal precision = false)	Data will be truncated at decimal places.	GENTYPE_NUMERIC_DOUBLE
decimal	Number (length <=18, decimal precision =true)	Data will be truncated at decimal places and other truncation results in Error.	GENTYPE_NUMERIC_PRECISION
short unsignedshort	Number (length <= 4)	Data will be truncated if it is > SHORT_MAX	GENTYPE_NUMERIC_SHORT
integer long int nonPositiveInteger nonNegativeInteger negativeInteger positiveInteger unsignedLong unsignedInt	Number (length <= 9)	Data will be truncated if it is > LONG_MAX	GENTYPE_NUMERIC_LONG

Note: Data map pairs which are not mentioned in the table are not considered for truncation. The following XSD data types are not truncated:

- boolean
- datetime
- Time
- Date
- gYearMonth
- gYear
- gMonthDay
- gDay
- gMonth
- Bytes

For example, when the web service response of XSD datetime data type are mapped to Text, the response is not truncated.

Arguments

None.

Return Code

A long value that represents one or more flags which specify that the web service response for the given data types are not truncated but instead an error is raised instead.

Default Behavior

If the web service response size is greater than the Gen data type, the web service response is truncated. For more information about the default behavior, see the ABRT XCall WS Gentype Truncation Table in the Purpose topic.

Building on Windows

The CALL EXTERNAL Web Service URL exit is built as part of the dynamic link library ABEXxxN.DLL.

Note: xx refers to the current release of CA Gen. For the current release number, see the Release Notes. N indicates platform.

Install the Microsoft Visual C++ compiler on your system as a prerequisite for building the DLL.

Follow these steps:

1. Launch a Command Prompt window.
2. Change your current directory to that directory which contains the makefile ABRTEXIT.NT. The path is %GENxx%Gen\VSabc for Visual Studio 32-bit or %GENxx%Gen\VSabc\amd64 for Visual Studio 64-bit.

   Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Set the Microsoft Visual Studio compiler environment variables.
4. Run NMAKE /F ABRTEXIT.NT.

Related User Exits

None.

UNIX and Linux Action Block Runtime User Exits

ABRT_xcall_ws_url_exit —CALL EXTERNAL Web Service URL Exit

```
void ABRT_xcall_ws_url_exit (char *url,
size_t urlMaxLen)
```

Source Code

ABRTEXIT.C

Purpose

The action block runtime invokes this user exit to let a user influence the web service URL associated with a CALL EXTERNAL web service action block statement.

The inputs for this user exit are the URL and the maximum size variables. The URL input variable is modified as necessary. However, the length of the modified URL must not exceed the specified urlMaxLen variable.

For more information about the input and output fields of this exit routine, see Arguments.

Arguments

The following table gives a brief description of each of the arguments.

Name	I/O	Description
*url	Input/Output	A pointer to a character array that contains the value of the web service URL. This user exit can set this value by modifying the data area pointed to by this argument.
urlMaxLen	Input	A size_t field that contains the maximum length allowed for the associated URL parameter.

Return Code

None

Default Behavior

The CALL EXTERNAL Web Service URL user exit, as delivered with CA Gen, will return without modifying the URL value.

Building on UNIX/Linux

This user exit is built as part of the shared library or archive library libae_userexits_c.*, where * is the shared library suffix or archive library depending on the UNIX system. As a prerequisite for building the shared library or archive library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f stuxexit.plat, where plat is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None.

ABRT_xcall_ws_gentype_truncate_exit —CALL EXTERNAL Data Truncation

```
long ABRT_xcall_ws_gentype_truncate_exit (void)
```

Source

ABRTEXIT.C

Purpose

By default, when the size of the web service response is greater than the matched Gen data type, the response is truncated. The truncation is governed by the table below.

Use this user exit to override the default behavior. The user exit sets the matched Gen data type flag so that the response is not truncated but an error is raised instead.

ABRT XCall WS Gentype Truncation Table:

The following table depicts the default behavior. The user exit sets the flag represented in the Overrides the Flag column. The default behavior overrides the flag.

XSD data type (from)	Gen data type (to)	Default truncation behavior	Overrides the flag
string	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
string	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB
anyURI	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
QName	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
NOTATION	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
duration	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
base64Binary	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
base64Binary	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB
hexBinary	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
hexBinary	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB

XSD data type (from)	Gen data type (to)	Default truncation behavior	Overrides the flag
float	Number (length <= 15, decimal points !=0, decimal precision = false)	Data will be truncated at decimal places.	GENTYPE_NUMERIC_DOUBLE
double	Number (length <= 15, decimal points !=0, decimal precision = false)	Data will be truncated at decimal places.	GENTYPE_NUMERIC_DOUBLE
decimal	Number (length <=18, decimal precision =true)	Data will be truncated at decimal places and other truncation results in Error.	GENTYPE_NUMERIC_PRECISION
short unsignedshort	Number (length <= 4)	Data will be truncated if it is > SHORT_MAX	GENTYPE_NUMERIC_SHORT
integer long int nonPositiveInteger nonNegativeInteger negativeInteger positiveInteger unsignedLong unsignedInt	Number (length <= 9)	Data will be truncated if it is > LONG_MAX	GENTYPE_NUMERIC_LONG

Note: Data map pairs which are not mentioned in the table are not considered for truncation. The following XSD data types are not truncated:

- boolean
- datetime
- Time
- Date
- gYearMonth
- gYear
- gMonthDay
- gDay
- gMonth
- Bytes

For example, when the web service response of XSD datetime data type are mapped to Text, the response is not truncated.

Arguments

None.

Return Code

A long value that represents one or more flags which specify that the web service response for the given data types are not truncated but instead an error is raised instead.

Default Behavior

If the web service response size is greater than the Gen data type, the web service response is truncated. For more information about the default behavior, see the ABRT XCall WS Gentype Truncation Table in the Purpose topic.

Building on UNIX/Linux

This user exit is built as part of the shared library or archive library libae_userexits_c.*, where * is the shared library suffix or archive library depending on the UNIX system. As a prerequisite for building the shared library or archive library, you must have correct C/C++ compiler installed on your system.

Follow these steps:

1. Launch a terminal window.
2. Change your current directory to that directory which contains the makefiles (by default, the $IEFH/make directory).
3. Run make /f stuxexit.plat, where plat is the matching platform extension:

   AIX: aix

   HP Itanium: ia64

   Solaris: sol

   Linux: lin

   The user exit shared library will be built in the $IEFH/lib directory.

Related User Exits

None.

Java Action Block Runtime User Exits

WebServiceMethodCallExit

Source Code

WebServiceMethodCallExit.java

Purpose

This user exit provides one method to modify the URL used to access a web service method at runtime and another method to modify the default truncation behavior used for return values during a web service call.

modifyURL Method-Modifies the URL

```
Public Static Final String modifyURL(String url)
```

Purpose

This method allows the URL used to access a web service method to be modified at runtime.

Arguments

The following table contains the arguments of the method:

Name	Output	Description
String	URL	The URL of the web service.

Return Code

The string representation of the URL of the web service.

Default Behavior

This method returns the URL of the web service that was added when the call external statement was created. The default behavior is to return the URL parameter unchanged.

ABRT_xcall_ws_gentype_truncate_exit —CALL EXTERNAL Data Truncation

```
long ABRT_xcall_ws_gentype_truncate_exit (void)
```

Purpose

By default, when the size of the web service response is greater than the matched Gen data type, the response is truncated. The truncation is governed by the table below.

Use this user exit to override the default behavior. The user exit sets the matched Gen data type flag so that the response is not truncated but an error is raised instead.

ABRT XCall WS Gentype Truncation Table:

The following table depicts the default behavior. The user exit sets the flag represented in the Overrides the Flag column. The default behavior overrides the flag.

XSD data type (from)	Gen data type (to)	Default truncation behavior	Overrides the flag
string	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
string	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB

XSD data type (from)	Gen data type (to)	Default truncation behavior	Overrides the flag
anyURI	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
QName	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
NOTATION	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
duration	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
base64Binary	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
base64Binary	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB
hexBinary	Text	Data truncated to fit in destination size.	GENTYPE_TEXT
hexBinary	BLOB	Data truncated to fit in destination size.	GENTYPE_BLOB
float	Number (length <= 15, decimal points !=0, decimal precision = false)	Data will be truncated at decimal places.	GENTYPE_NUMERIC_DOUBLE
double	Number (length <= 15, decimal points !=0, decimal precision = false)	Data will be truncated at decimal places.	GENTYPE_NUMERIC_DOUBLE
decimal	Number (length <=18, decimal precision =true)	Data will be truncated at decimal places and other truncation results in Error.	GENTYPE_NUMERIC_PRECISION
short unsignedshort	Number (length <= 4)	Data will be truncated if it is > SHORT_MAX	GENTYPE_NUMERIC_SHORT

XSD data type (from)	Gen data type (to)	Default truncation behavior	Overrides the flag
integer long int nonPositiveInteger nonNegativeInteger negativeInteger positiveInteger unsignedLong unsignedInt	Number (length <= 9)	Data will be truncated if it is > LONG_MAX	GENTYPE_NUMERIC_LONG

Note: Data map pairs which are not mentioned in the table are not considered for truncation. The following XSD data types are not truncated:

- boolean
- datetime
- Time
- Date
- gYearMonth
- gYear
- gMonthDay
- gDay
- gMonth
- Bytes

For example, when the web service response of XSD datetime data type are mapped to Text, the response is not truncated.

Arguments

None.

Return Code

A long value that represents one or more flags which specify that the web service response for the given data types are not truncated but instead an error is raised instead.

Default Behavior

If the web service response size is greater than the Gen data type, the web service response is truncated. For more information about the default behavior, see the ABRT XCall WS Gentype Truncation Table in the Purpose topic.

Building on Windows

This exit is compiled into a Java class file. The Java development environment Java Platform SE 1.6 or higher must be installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Ensure that the CLASSPATH environment variable contains a reference to the CA Gen CSU jar file, CSUxx.JAR.

   Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
3. Change your current directory to that which contains the java source file.

   Typically this is in the CA Gen install area, within the CLASSES\COM\CA\GENxx\EXITS\Common subdirectory.

   Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.
4. Run javac WebServiceMethodCallExit.java.
5. Reassemble and redeploy the resulting .class file for use with the appropriate applications.

Related User Exits

None.