ORACLE
PEOPLESOFT ENTERPRISE

# PeopleTools 8.51 PeopleBook: PeopleSoft MultiChannel Framework

**August 2010**

ORACLE

PeopleTools 8.51 PeopleBook: PeopleSoft MultiChannel Framework
SKU pt8.51tmcf-b0810

Copyright © 1988, 2010, Oracle and/or its affiliates. All rights reserved.

**Trademark Notice**

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

**License Restrictions Warranty/Consequential Damages Disclaimer**

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

**Warranty Disclaimer**

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

**Restricted Rights Notice**

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

*U.S. GOVERNMENT RIGHTS*

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

**Hazardous Applications Notice**

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

**Third Party Content, Products, and Services Disclaimer**

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products or services.

# Contents

**Preface**

**PeopleSoft MultiChannel Framework Preface ..... xxxiii**

PeopleSoft MultiChannel Framework ..... xxxiii
Additional Resources ..... xxxiii
PeopleBooks and the PeopleSoft Online Library ..... xxxiii

**Chapter 1**

**Getting Started with PeopleSoft MultiChannel Framework ..... 1**

PeopleSoft MultiChannel Framework Overview ..... 1
PeopleSoft MultiChannel Framework Implementation ..... 1

**Chapter 2**

**Understanding PeopleSoft MultiChannel Framework ..... 5**

PeopleSoft MultiChannel Framework ..... 5
PeopleSoft MultiChannel Framework Elements and Channels ..... 5
PeopleSoft MultiChannel Framework Elements ..... 6
PeopleSoft MultiChannel Framework Channels ..... 6
PeopleSoft MultiChannel Framework Universal Queue ..... 7
PeopleSoft MultiChannel Console ..... 8
PeopleSoft MCF Architecture ..... 8
PeopleSoft MCF Server Architecture ..... 8
Chat Architecture ..... 9
Email Architecture ..... 11

**Chapter 3**

**Configuring PeopleSoft Computer Telephony Integration ..... 15**

Understanding the PeopleSoft CTI Console ..... 15
PeopleSoft CTI ..... 15
PeopleSoft CTI Components ..... 16
Adapter-Based CTI Requirements ..... 17

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

PeopleSoft MultiChannel API ..... 18
Required Security for PSMCAPI ..... 20
JavaScript MultiChannel API ..... 20
Applet-Based CTI Requirements ..... 20
Java Applet Console ..... 20
Java Runtime Environment ..... 21
Configuring PeopleSoft CTI ..... 22
Pages Used to Configure CTI ..... 22
Installing PeopleSoft CTI ..... 23
Enabling PeopleSoft CTI ..... 24
Configuring CTI Console Type ..... 24
Creating a List of Frequently Dialed Phone Numbers ..... 25
Entering Default Screen Pop-Up URL ..... 26
Using the Reason Code Page ..... 27
Configuring PeopleSoft CTI Using Adapters ..... 28
Page Used to Configure the CTI Non-Applet Console ..... 28
Configuring the CTI Non-Applet Console ..... 29
Configuring PeopleSoft CTI Using Applets ..... 29
Pages Used to Configure CTI Using Applets ..... 30
Configuring the CTI Applet Console ..... 30
Using the CTI Genesys Page ..... 32
Using the CTI Cisco Page ..... 33
Configuring PeopleSoft CTI Queues and CTI Agents ..... 36
Pages Used to Configure CTI Queues and CTI Agents ..... 37
Configuring CTI Queues ..... 37
Configuring CTI Agents ..... 38
Using the Phone Book Page ..... 40
Personalizing the Agent Console ..... 41
Viewing Information About the Agent Information Page ..... 43
Using Other PeopleSoft CTI Options ..... 44
Configuring Pop-Up Windows ..... 44
Supporting Single Sign-In ..... 47
Logging CTI Events ..... 47
Implementing Free Seating ..... 47
Using the PeopleSoft CTI Sample Pages ..... 47
Page Used to Demonstrate Outbound Calls ..... 48
Using the Outbound Call Page ..... 48

**Chapter 4**

**Using PeopleSoft CTI ..... 51**

Understanding PeopleSoft CTI ..... 51
PeopleSoft CTI ..... 51


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Cisco Switch Considerations for the Applet-Based Solution ..... 52
Using PeopleSoft CTI ..... 52
Getting Started ..... 53
Using the CTI Console ..... 55
Selecting Call Actions ..... 56
Answering a Call ..... 58
Transferring a Caller ..... 58
Initiating Conference Calls ..... 60
Working with the Hold Status ..... 61
Disconnecting a Caller ..... 61
Switching Agent Ready Status ..... 61
Dialing an Outbound Call ..... 62
Completing a Call ..... 63
Using Hot Keys ..... 63

**Chapter 5**

**Configuring REN Servers ..... 65**

Understanding REN Servers ..... 65
REN Server Failover, Scalability, and Security Configuration ..... 65
REN Server Failover ..... 67
REN Server Clusters ..... 67
Understanding SSL-Enabled REN Servers ..... 68
Installing Digital Certificates ..... 68
Authenticating Server and Client ..... 68
Performance and Scalability for SSL-Enabled REN Servers ..... 69
Configuring REN Server Security ..... 69
Understanding REN Server Security Configuration ..... 69
Defining Permission Lists for REN Server Access ..... 69
Configuring REN Servers ..... 71
Understanding REN Server Configuration Options ..... 71
Page Used to Configure REN Servers ..... 74
Configuring REN Servers and SSL-Enabled REN Servers ..... 74
Defining REN Servers ..... 77
Configuring REN Server and SSL-Enabled REN Server Clusters ..... 80
Pages Used to Configure REN Server and SSL-Enabled REN Server Clusters ..... 80
Defining a REN Server Cluster ..... 80
Specifying REN Server Ownership ..... 83
Specifying REN Server Cluster Members ..... 84
Configuring a Reverse Proxy Server with a REN Server ..... 84
Understanding RPS Configuration ..... 85
Example: Configuring a WebLogic RPS for a REN Server on Another Host Machine ..... 85
Configuring Apache-based Reverse Proxy Servers for a REN Server ..... 87

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Chapter 6**

**Configuring PeopleSoft MCF Servers and Clusters ..... 89**

Understanding PeopleSoft MCF Server and Cluster Architecture ..... 89
PeopleSoft MCF Server Configuration ..... 89
PeopleSoft MCF Cluster Architecture ..... 90
Queue Server and Queue Server Failover ..... 92
Logical Queues and Physical Queues ..... 93
PeopleSoft MCF Log Server and Log Server Failover ..... 93
Queue Server Scalability ..... 93
Recommended Configurations ..... 94
Configuring PeopleSoft MCF Clusters ..... 94
Understanding PeopleSoft MCF Cluster Configuration ..... 95
Configuring PeopleSoft MCF Clusters ..... 95

**Chapter 7**

**Configuring PeopleSoft MCF Queues and Tasks ..... 97**

Defining Queues ..... 97
Pages Used to Define Queues ..... 97
Defining Queues ..... 98
Defining Chat Responses ..... 99
Defining Static Push URLs ..... 100
Configuring Tasks ..... 101
Page Used to Configure Tasks ..... 102
Configuring Tasks ..... 102
Viewing the Cluster Summary ..... 107
Page Used to View the Cluster Summary ..... 107
Viewing the Cluster Summary ..... 107
Tuning Cluster Parameters ..... 108
Page Used to Tune Clusters ..... 108
Tuning Cluster Parameters ..... 108
Notifying Clusters of Changed Parameters ..... 116
Page Used to Notify Clusters of Changed Parameters ..... 116
Notifying Clusters of Changed Parameters ..... 117

**Chapter 8**

**Configuring PeopleSoft MCF Agents ..... 119**

Defining Agents ..... 119


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Pages Used to Define Agents ..... 119
Creating Agents ..... 120
Specifying Languages That an Agent Supports ..... 121
Personalizing an Agent's Presence ..... 122
Defining Optional Agent Characteristics ..... 123
Pages Used to Define Optional Agent Characteristics ..... 124
Setting Up Buddy Lists ..... 124
Configuring Windows ..... 125
Personalizing Chat ..... 127
Specifying Agent-Specific URLs ..... 128
Specifying Miscellaneous Parameters ..... 129

**Chapter 9**

**Administering Queues, Logs, and Tasks ..... 133**

Administering Physical Queues ..... 133
Pages Used to Administer Physical Queues ..... 133
Moving Agents Between Physical Queues ..... 134
Moving Queues ..... 135
Balancing Queues ..... 136
Viewing Queue Server, Queue, and Agent States ..... 137
Pages Used to View Queue Server, Queue, and Agent States ..... 137
Viewing the Queue Server State ..... 137
Viewing the Queue State Summary ..... 138
Viewing the Agent State Summary ..... 139
Viewing Broadcast, Chat, and Event Logs ..... 140
Pages Used to View Broadcast, Chat, and Event Logs ..... 141
Viewing Broadcast Logs ..... 141
Viewing Chat Logs ..... 142
Viewing Event Logs ..... 144
Viewing PeopleSoft MCF Logs ..... 147
Administering Overflow and Escalated Tasks ..... 148
Pages Used to Administer Overflow and Escalated Tasks ..... 148
Administering Overflow Tasks ..... 148
Administering Escalated Tasks ..... 149

**Chapter 10**

**Managing Tasks and Using Chat in PeopleSoft MultiChannel Framework ..... 151**

Managing Tasks with the MultiChannel Console ..... 151
Using the MultiChannel Console to Work with Tasks ..... 151
Communicating with Customers and Agents Using Chat ..... 153

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Using the Agent Chat Window ..... 153
Using the Customer Chat Window ..... 155

**Chapter 11**

**Using PeopleSoft MCF Broadcast and Working with Sample Pages ..... 159**

Using PeopleSoft MCF Broadcast ..... 159
Pages Used to Configure PeopleSoft MCF Broadcast ..... 160
Understanding JSMCAPI Broadcast ..... 160
Implementing MCF Broadcast ..... 161
Using JSMCAPI Broadcast with MCF Supervisor Console ..... 161
Using PeopleCode Broadcast ..... 166
Viewing Broadcast Logs ..... 169
Working with Sample Pages ..... 169
Pages Used to Work with Sample Pages ..... 170
Using the Customer Chat Sample Page ..... 170
Using the URL Wizard ..... 172
Using the Generic Event Sample Page ..... 173
Using the Generic Event Window ..... 175
Using the Email Sample Page ..... 176
Using the Email Window ..... 177
Using and Demonstrating JSMCAPI ..... 178
Understanding JSMCAPI ..... 178
Common Elements Used in This Section ..... 183
Pages Used to Use and Demonstrate JSMCAPI ..... 184
Using the CTI Sample Console ..... 184
Using the Agent Console Page ..... 192
Using the Monitor Agents Page and Sample Monitor - Agent States Page ..... 195
Using the Monitor Queues Page and Sample Monitor - Queue Statistics Page ..... 197
Using PeopleCode Built-in Functions with PeopleSoft MultiChannel Framework ..... 201
Using Universal Queue Classes ..... 202

**Chapter 12**

**Configuring PeopleSoft MCF for Third-Party Routing Systems ..... 203**

Understanding Third-Party Routing Systems ..... 204
Defining Third-Party Routing System Requirements ..... 207
Defining Third-Party Routing Rules ..... 207
Configuring PeopleSoft MCF for a Third Party ..... 208
Defining the Third-Party Flag ..... 209
Page Used to Define the Third-Party Flag ..... 209
Defining the Third-Party Flag ..... 209


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Defining the PeopleSoft MCF Cluster Page for a Third Party ..... 210
Page Used to Define the PeopleSoft MCF Cluster Page for a Third Party ..... 210
Defining the PeopleSoft MCF Cluster Page for a Third Party ..... 210
Tuning PeopleSoft MCF Cluster Parameters for a Third Party ..... 212
Page Used to Tune PeopleSoft MCF Cluster Parameters for a Third Party ..... 212
Tuning PeopleSoft MCF Cluster Parameters for a Third Party ..... 212
Notifying PeopleSoft MCF Clusters of Changed Parameters for a Third Party ..... 215
Page Used to Notify Third-Party Clusters of Changed Parameters ..... 216
Notifying Third-Party Clusters of Changed Parameters ..... 216
Defining PeopleSoft MCF Queues for a Third Party ..... 217
Pages Used to Define PeopleSoft MCF Queues for a Third Party ..... 217
Defining PeopleSoft MCF Queues for a Third Party ..... 217
Defining Canned Queue Messages ..... 218
Defining Canned Queue URLs ..... 219
Defining PeopleSoft MCF Agents for a Third Party ..... 220
Pages Used to Define PeopleSoft MCF Agents for a Third Party ..... 221
Creating PeopleSoft MCF Agents for a Third Party ..... 222
Setting Up Buddy Lists ..... 223
Customizing Windows ..... 223
Defining Messages ..... 225
Specifying Agent-Specific URLs ..... 226
Defining Agent's Presence ..... 227
Specifying the Media ..... 228
Specifying Languages That an Agent Supports ..... 229
Specifying Miscellaneous Parameters ..... 230
Configuring PeopleSoft MCF Tasks for a Third Party ..... 231
Page Used to Configure PeopleSoft MCF Task for a Third Party ..... 232
Configuring PeopleSoft MCF Task for a Third Party ..... 232
Configuring CTI for a Third Party ..... 232
Communicating with Customers and Agents Using Chat ..... 232
Using the Third-Party Chat Window ..... 232
Using the Customer Chat Window ..... 234
Viewing Event Logs for a Third Party ..... 236
Page Used to View Event Logs for a Third Party ..... 236
Viewing Event Logs for a Third Party ..... 236
Viewing Broadcast Logs for a Third Party ..... 238
Page Used to View Broadcast Logs for a Third Party ..... 238
Viewing Broadcast Logs for a Third Party ..... 238
Working with Third-Party Sample Pages ..... 239
Pages Used to Work with Third-Party Sample Pages ..... 239
Using the Customer Chat Sample Page ..... 240
Using the Generic Event Sample Page ..... 240
Using the Email Sample Page ..... 240
Using the Sample Console Page ..... 240
Using the MCF Broadcast Page ..... 249

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Using the PCodeBroadcast Page ..... 250

**Chapter 13**

**Understanding JSMCAPI Classes ..... 251**

Understanding JSMCAPI ..... 251
Understanding JSMCAPI Classes ..... 251
PSMC ..... 255
Server ..... 258
RENServer ..... 258
Session ..... 258
_Address ..... 258
Line ..... 259
Connection ..... 259
Group ..... 259
Task ..... 260
_User ..... 260
MediaType ..... 260
Reason ..... 261
Statistics ..... 261
Data ..... 262
Globals ..... 262
MCEvent ..... 262
Caps ..... 262
ForwardMode ..... 263
_Address Class Hierarchy ..... 263
_Address Class Constructor ..... 264
_Address ..... 265
_Address Class Fields ..... 265
caps ..... 265
id ..... 265
_Address Class Callback Event Method ..... 266
onError ..... 266
_UQAddress Class Constructor ..... 266
_UQAddress ..... 266
_UQAddress Class Fields ..... 267
Tasks ..... 267
_UQAddress Methods ..... 267
acceptTask ..... 267
dequeueTask ..... 268
_UQAddress Class Callback Event Methods ..... 268
onAccepted ..... 268
onAcceptingTask ..... 269


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

onDequeueingTask ..... 269
onNotify ..... 269
onTaskAdded ..... 269
onTaskRemoved ..... 270
onUnassigned ..... 270
_User Class Hierarchy ..... 270
_User Class Constructor ..... 271
_User ..... 271
_User Class Fields ..... 271
agentID ..... 271
caps ..... 272
id ..... 272
name ..... 272
presences ..... 272
states ..... 272
ST_LOGGEDIN ..... 273
ST_LOGGEDOUT ..... 273
ST_NOTREADY ..... 273
ST_READY ..... 273
ST_UNKNOWN ..... 273
ST_WORKNOTREADY ..... 273
ST_WORKREADY ..... 274
statistics ..... 274
statistics1 ..... 274
statistics2 ..... 274
A2AChat Class Constructor ..... 275
A2AChat ..... 275
A2AChat Class Fields ..... 275
address ..... 276
agentID ..... 276
agentName ..... 276
appData ..... 276
chatType ..... 277
customerName ..... 277
id ..... 277
isConference ..... 278
jr ..... 278
question ..... 278
subject ..... 278
type ..... 278
uniqueueId ..... 279
A2AChat Class Method ..... 279
getURL ..... 279
A2AChatAddress Class Constructor ..... 279
A2AChatAddress ..... 280

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

A2AChatAddress Class Fields ..... 280
id ..... 280
tasks ..... 281
A2AChatAddress Class Method ..... 281
initiateChat ..... 281
A2AChatAddress Class Callback Event Methods ..... 282
onChatEnded ..... 282
onInitiatingChat ..... 282
onNotify ..... 282
AgentStatistics Class Constructor ..... 283
AgentStatistics ..... 283
AgentStatistics Class Fields ..... 283
averageCallDuration ..... 283
averageHoldDuration ..... 284
callsHandled ..... 284
data ..... 284
percentIdleTime ..... 284
percentTimeAvailable ..... 284
percentTimeInCurrentState ..... 285
percentTimeUnavailable ..... 285
timeCurrentLogin ..... 285
timeWorking ..... 285
totalTaskAcceptedLogin ..... 285
totalTaskDoneLogin ..... 286
totalTaskUnassignedLogin ..... 286
unavailableDuration ..... 286
waitDuration ..... 286
AppData Class Constructor ..... 286
AppData ..... 287
AppData Class Fields ..... 287
data ..... 287
groupId ..... 287
jr ..... 288
question ..... 288
strData ..... 288
subject ..... 288
uniqueId ..... 288
url ..... 289
userId ..... 289
username ..... 289
wizUrl ..... 289
AppData Class Method ..... 289
addKeyValue ..... 290
Buddy Class Constructor ..... 290
Buddy ..... 290


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Buddy Class Fields ..... 291
Buddy Class Callback Event Methods ..... 291
onStat1 ..... 291
onStat2 ..... 292
onState ..... 292
Call Class Constructor ..... 292
Call ..... 292
Call Class Fields ..... 293
line ..... 293
statistics ..... 294
CallData Class Constructor ..... 294
CallData ..... 294
CallData Class Fields ..... 295
ani ..... 295
callId ..... 295
callType ..... 295
data ..... 295
dnis ..... 296
CallData Class Method ..... 296
addKeyValue ..... 296
CallStatistics Class Constructor ..... 296
CallStatistics ..... 297
CallStatistics Class Fields ..... 297
data ..... 297
holdTime ..... 297
queueTime ..... 298
talkTime ..... 298
Chat Class Constructor ..... 298
Chat ..... 298
Chat Class Fields ..... 299
address ..... 299
agentId ..... 300
appData ..... 300
chatconnection ..... 300
chatType ..... 300
customerName ..... 301
groupId ..... 301
question ..... 301
subject ..... 301
statistics ..... 301
userData ..... 302
Chat Class Method ..... 302
gettpUrl ..... 302
getUrl ..... 303
ChatAddress Class Constructor ..... 303

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

ChatAddress ..... 303
ChatAddress Class Fields ..... 304
chatconnections ..... 304
ChatAddress Class Methods ..... 304
chat ..... 305
getChatconnectionByConnectionId ..... 305
getChatconnectionindexByConnectionId ..... 306
getFreeChatconnection ..... 306
getFreeChatconnectionIndex ..... 307
ChatAddress Callback Event Methods ..... 307
onCapabilitiesChanged ..... 307
ChatConnection Class Constructor ..... 308
ChatConnection ..... 308
ChatConnection Class Fields ..... 308
caps ..... 308
chat ..... 309
connectionId ..... 309
id ..... 309
state ..... 309
ChatConnection Class Methods ..... 310
answer ..... 310
attachUserData ..... 310
conference ..... 311
forward ..... 311
gethistory ..... 312
getUrl ..... 313
message ..... 313
pushURL ..... 314
reject ..... 314
release ..... 315
typing ..... 315
wrapup ..... 316
ChatConnection Class Callback Event Methods ..... 317
onAccepted ..... 317
onAnswering ..... 317
onCapabilitiesChanged ..... 317
onChatdataChanged ..... 318
onConferencing ..... 318
onDialing ..... 318
onDropped ..... 318
onError ..... 319
onForwarded ..... 319
onForwardError ..... 319
onForwarding ..... 319
onHistory ..... 320


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

onIncomingChat ..... 320
onMessage ..... 320
onPartyAdded ..... 320
onPartyChanged ..... 321
onPartyRemoved ..... 321
onProperties ..... 321
onPushURL ..... 321
onRejected ..... 322
onReleased ..... 322
onReleasing ..... 322
onRevoked ..... 322
onTalking ..... 323
onTyping ..... 323
onUserdataChanged ..... 323
ChatConnectionCaps Class Constructor ..... 323
ChatConnectionCaps ..... 324
ChatConnectionCaps Class Fields ..... 324
canAnswer ..... 324
canConference ..... 324
canConferenceSingle ..... 325
canForward ..... 325
canIndicateTyping ..... 325
canPushURL ..... 325
canReject ..... 325
canSendMessage ..... 326
ChatData Class Constructor ..... 326
ChatData ..... 326
ChatData Class Fields ..... 326
data ..... 326
ChatData Class Methods ..... 326
addKeyValue ..... 327
Email Class Constructor ..... 327
Email ..... 327
Email Class Fields ..... 328
address ..... 328
agentId ..... 329
appData ..... 329
customerName ..... 329
emailconnection ..... 329
emailId ..... 329
groupId ..... 330
question ..... 330
statistics ..... 330
subject ..... 330
userData ..... 330

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Email Class Method ..... 331
gettpUrl ..... 331
getUrl ..... 331
EmailAddress Class Constructor ..... 332
EmailAddress ..... 332
EmailAddress Class Fields ..... 332
agent ..... 333
emailconnections ..... 333
EmailAddress Class Methods ..... 333
getEmailconnectionByConnectionId ..... 333
getEmailconnectionindexByConnectionId ..... 334
getFreeEmailconnection ..... 334
EmailAddress Callback Event Methods ..... 335
EmailConnection Class Constructor ..... 335
EmailConnection ..... 335
EmailConnection Class Fields ..... 335
caps ..... 336
connectionId ..... 336
email ..... 336
id ..... 336
state ..... 336
EmailConnection Class Methods ..... 337
abandon ..... 337
answer ..... 337
attachUserData ..... 338
complete ..... 339
forward ..... 339
reject ..... 340
withdraw_RES ..... 340
EmailConnection Class Callback Event Methods ..... 341
onAnswering ..... 341
onCapabilitiesChanged ..... 341
onCompleted ..... 341
onDropped ..... 342
onEmaildataChanged ..... 342
onError ..... 342
onForwarded ..... 342
onForwardError ..... 343
onForwarding ..... 343
onIncoming ..... 343
onProcessing ..... 343
onRejected ..... 344
onRevoked ..... 344
onUserdataChanged ..... 344
onWithdraw_REQ ..... 344


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

EmailConnectionCaps Class Constructor ..... 345
EmailConnectionCaps ..... 345
EmailConnectionCaps Class Fields ..... 345
canAnswer ..... 345
canComplete ..... 346
canForward ..... 346
canReject ..... 346
EmailData Class Constructor ..... 346
EmailData ..... 346
EmailData Class Fields ..... 347
data ..... 347
EmailData Class Methods ..... 347
addKeyValue ..... 347
Extension Class Constructor ..... 348
Extension ..... 348
Extension Class Fields ..... 348
forwardMode ..... 349
isDnd ..... 349
lines ..... 349
numOfLines ..... 349
Extension Class Methods ..... 349
cancelDnd ..... 350
cancelForwardSet ..... 350
forwardSet ..... 351
getDialingLine ..... 351
getFreeLine ..... 352
getLineByConnectionId ..... 352
getOffHookLine ..... 353
setDnd ..... 353
Extension Class Callback Event Methods ..... 354
onCancelingDnd ..... 354
onCancelingForward ..... 354
onDnd ..... 354
onDndCanceled ..... 355
onForwardCanceled ..... 355
onForwarded ..... 355
onForwarding ..... 356
onSettingDnd ..... 356
ExtensionCaps Class Constructor ..... 356
ExtensionCaps ..... 356
ExtensionCaps Class Fields ..... 357
canCancelDnd ..... 357
canDial ..... 357
canFwdBusy ..... 357
canFwdBusyNoAnswer ..... 358

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

canFwdCancelForward ..... 358
canFwdDefault ..... 358
canFwdNoAnswer ..... 358
canFwdUnconditional ..... 358
canRefreshState ..... 359
canSetDnd ..... 359
ForwardMode Class Constructor ..... 359
ForwardMode ..... 359
ForwardMode Class Field ..... 360
mode ..... 360
GenericAddress Class Constructor ..... 360
GenericAddress ..... 360
GenericAddress Class Fields ..... 361
agent ..... 361
genericconnections ..... 361
GenericAddress Class Methods ..... 362
getFreeGenericconnection ..... 362
getGenericconnectionByConnectionId ..... 362
getGenericconnectionindexByConnectionId ..... 363
GenericAddress Class Callback Event Methods ..... 363
GenericConnection Class Constructor ..... 364
GenericConnection ..... 364
GenericConnection Class Fields ..... 364
caps ..... 365
connectionId ..... 365
generic ..... 365
id ..... 365
state ..... 365
GenericConnection Class Methods ..... 366
abandon ..... 366
answer ..... 366
attachUserData ..... 367
complete ..... 368
forward ..... 368
reject ..... 369
withdraw_RES ..... 369
GenericConnection Class Callback Event Methods ..... 370
onCapabilitiesChanged ..... 370
onCompleted ..... 370
onDropped ..... 370
onError ..... 371
onForwarded ..... 371
onForwardError ..... 371
onForwarding ..... 371
onGenericdataChanged ..... 372


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

onIncoming ..... 372
onProcessing ..... 372
onRejected ..... 372
onRevoked ..... 373
onUserdataChanged ..... 373
onWithdraw_REQ ..... 373
GenericConnectionCaps Class Constructor ..... 373
GenericConnectionCaps ..... 374
GenericConnectionCaps Class Fields ..... 374
canAnswer ..... 374
canComplete ..... 374
canForward ..... 375
canReject ..... 375
GenericData Class Constructor ..... 375
GenericData ..... 375
GenericData Class Fields ..... 375
data ..... 376
GenericData Class Methods ..... 376
addKeyValue ..... 376
GenericTask Class Constructor ..... 376
GenericTask ..... 377
GenericTask Class Fields ..... 377
address ..... 378
agentId ..... 378
appData ..... 378
customerName ..... 378
genericconnection ..... 379
genericId ..... 379
groupId ..... 379
question ..... 379
statistics ..... 379
subject ..... 380
userdata ..... 380
GenericTask Class Method ..... 380
gettpUrl ..... 380
getUrl ..... 381
GLOBALS Class Fields ..... 381
A2AChat.PS_JR ..... 381
A2AChat.TYPE_ANSWER ..... 382
A2AChat.TYPE_CONSULT ..... 382
Server.TYPE_CTI ..... 382
Server.TYPE_UQ ..... 382
Task.TYPE_A2ACHAT ..... 383
Task.TYPE_CHAT ..... 383
Task.TYPE_CTI ..... 383

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Task.TYPE_EMAIL ..... 383
Task.TYPE_GENERIC ..... 384
GLOBALS Class Methods ..... 384
initJSMCAPI ..... 384
isValid ..... 385
MCFBroadcast ..... 385
Group Class Constructor ..... 386
Group ..... 386
Group Class Fields ..... 387
id ..... 387
name ..... 387
registered ..... 387
statistics ..... 387
statistics1 ..... 388
statistics2 ..... 388
Group Class Callback Event Methods ..... 388
onStat ..... 388
onStat1 ..... 389
onStat2 ..... 389
onTaskAdded ..... 389
onTaskRemoved ..... 389
GroupStatistics Constructor ..... 390
GroupStatistics ..... 390
GroupStatistics Fields ..... 390
data ..... 390
listOfTasksInTheQueueByTaskType ..... 390
maxTaskCompletionTime ..... 391
newestTask ..... 391
newestTaskCompletionTime ..... 391
numberOfAbandoned ..... 391
numberOfLoggedIn ..... 391
numberOfQueued ..... 391
numUnassignedTasks ..... 392
queuedWaitTime ..... 392
queueUpTime ..... 392
relativeQueueLoad ..... 392
timeElapsedOldestTask ..... 392
GroupStatistics1 Class Constructor ..... 392
GroupStatistics1 ..... 393
GroupStatistics1 Class Fields ..... 393
mostRecentTaskDone ..... 393
mostRecentTaskDoneData ..... 393
mostRecentTaskEnqueued ..... 394
mostRecentTaskEnqueuedData ..... 394
numAgentsAvailable ..... 394


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

numAgentsLoggedIn ..... 394
numEscalation ..... 394
numOverflow ..... 395
numTaskAccepted ..... 395
numTaskDone ..... 395
numTaskQueued ..... 395
reasonFlag ..... 395
taskTotalTimeInSystem ..... 396
timeSinceStart ..... 396
GroupStatistics2 Class Constructor ..... 396
GroupStatistics2 ..... 396
GroupStatistics2 Class Fields ..... 397
averageTaskDuration ..... 397
averageWaitTime ..... 397
oldestTask ..... 397
recentTask ..... 397
timeElapsedOldestTask ..... 398
timeElapsedRecentTask ..... 398
Line Class Constructor ..... 398
Line ..... 398
Line Class Fields ..... 399
call ..... 399
caps ..... 399
connectionid ..... 399
id ..... 400
isMuted ..... 400
state ..... 400
Line Class Methods ..... 401
alternate ..... 401
answer ..... 401
attachUserData ..... 402
clear ..... 402
complete ..... 403
conference ..... 403
conferenceSingle ..... 404
dial ..... 405
dropParty ..... 405
getAni ..... 406
getDescr ..... 406
getDnis ..... 407
getPadvalue ..... 407
getReferenceId ..... 407
getUrl ..... 408
grabCall ..... 408
hold ..... 409

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

join ..... 409
mute ..... 410
park ..... 411
reconnect ..... 411
reject ..... 412
release ..... 412
retrieve ..... 413
sendDTMF ..... 413
setcallresult ..... 414
setcallresultDNC ..... 414
setcallresultReschedule ..... 415
transfer ..... 416
transferMute ..... 416
unmute ..... 417
updateCallData ..... 417
Line Class Callback Event Methods ..... 418
onAlternating ..... 418
onAnswering ..... 418
onAttachingUD ..... 419
onCallDataChanged ..... 419
onCapabilitiesChanged ..... 419
onClearing ..... 419
onCompleting ..... 420
onConferencing ..... 420
onDialing ..... 420
onDropped ..... 420
onError ..... 421
onGrabbing ..... 421
onHeld ..... 421
onHolding ..... 421
onJoining ..... 422
onMuted ..... 422
onOffHook ..... 422
onOnHook ..... 422
onParking ..... 423
onPartyAdded ..... 423
onPartyChanged ..... 423
onPartyRemoved ..... 423
onReconnecting ..... 424
onRejected ..... 424
onRejecting ..... 424
onReleasing ..... 424
onRetrieving ..... 425
onRinging ..... 425
onSetcallresult ..... 425


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

onSetcallresultDNC ..... 425
onSetcallresultReschedule ..... 426
onTalking ..... 426
onTransfering ..... 426
onUnmuted ..... 426
onUpdatingCD ..... 427
onUserDataChanged ..... 427
LineCaps Class Constructor ..... 427
LineCaps ..... 427
LineCaps Class Fields ..... 428
canAlternate ..... 428
canAnswer ..... 428
canAttachUserDate ..... 428
canClear ..... 428
canComplete ..... 429
canConference ..... 429
canConferenceSingle ..... 429
canDropParty ..... 429
canHold ..... 429
canMute ..... 430
canPark ..... 430
canReconnect ..... 430
canReject ..... 430
canRelease ..... 430
canRetrieve ..... 431
canSendDTMF ..... 431
canSetcallresult ..... 431
canSetcallresultDNC ..... 431
canSetcallresultReschedule ..... 431
canTransfer ..... 432
canTransferMute ..... 432
canUnmute ..... 432
canUpdateCallData ..... 432
MCEvent Class Constructor ..... 432
MCEvent ..... 433
MCEvent Class Fields ..... 433
extension ..... 433
group ..... 433
reason ..... 434
user ..... 434
MediaType Class Constructor ..... 434
MediaType ..... 434
MediaType Class Field ..... 435
type ..... 435
PSMC Class Constructor ..... 435

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

PSMC ..... 436
PSMC Class Fields ..... 436
renserver ..... 436
servers ..... 437
sessions ..... 437
PSMC Class Methods ..... 437
closeSession ..... 437
getCallById ..... 438
getChatById ..... 438
getEmailById ..... 439
getGenericTaskById ..... 439
getLineById ..... 440
openSession ..... 441
start ..... 441
stop ..... 442
Reason Class Constructor ..... 442
Reason ..... 442
Reason Class Fields ..... 443
code ..... 443
desc ..... 443
reasonData1 ..... 443
reasonData2 ..... 444
reasonData3 ..... 444
RenServer Class Constructor ..... 444
RenServer ..... 444
RenServer Class Fields ..... 445
isRunning ..... 445
url ..... 445
RenServer Class Callback Event Methods ..... 445
onDown ..... 445
onUp ..... 446
Server Class Constructor ..... 446
Server ..... 446
Server Class Fields ..... 446
id ..... 447
info ..... 447
state ..... 447
type ..... 447
Server Class Callback Event Methods ..... 448
onBroadcast ..... 448
onHbLost ..... 448
onHbRecovered ..... 449
onInService ..... 449
onOutOfService ..... 449
onRestart ..... 449


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Session Class Constructor ..... 450
Session ..... 450
Session Class Fields ..... 450
addresses ..... 450
buddies ..... 451
groups ..... 451
id ..... 451
intervalBetweenReqs ..... 451
numberRegsPerBulkReq ..... 451
serverId ..... 452
state ..... 452
user ..... 452
Session Class Methods ..... 452
broadcastSubscribe ..... 453
broadcastUnsubscribe ..... 453
close ..... 453
open ..... 454
registerAddress ..... 454
registerBuddy ..... 455
registerBuddiesBulk ..... 455
registerGroup ..... 456
registerGroupsBulk ..... 456
registerUser ..... 457
setAutoRecovery ..... 457
statPublish ..... 458
unregisterAddress ..... 458
unregisterBuddy ..... 459
unregisterGroup ..... 459
unregisterUser ..... 460
Session Class Callback Event Methods ..... 460
onAddressRegistered ..... 461
onAddressUnregistered ..... 461
onBuddyRegistered ..... 461
onBuddyUnregistered ..... 461
onClosed ..... 462
onError ..... 462
onGroupRegistered ..... 462
onGroupUnregistered ..... 462
onInfo ..... 463
onOpened ..... 463
onUserRegistered ..... 463
onUserUnregistered ..... 463
Task Class Hierarchy ..... 464
Task Class Constructor ..... 464
Task ..... 464

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Task Class Fields ..... 465
caseid ..... 465
cost ..... 465
customerid ..... 465
group ..... 466
id ..... 466
onStat ..... 466
priority ..... 466
type ..... 466
urlAbs ..... 467
urlRel ..... 467
TaskStatistics Class Constructor ..... 467
TaskStatistics ..... 468
TaskStatistics Class Fields ..... 468
data ..... 468
holdTime ..... 468
queueTime ..... 468
talkTime ..... 469
User Class Constructor ..... 469
User ..... 469
User Class Fields ..... 470
addresses ..... 470
agentPassword ..... 471
language ..... 471
registeredAddresses ..... 471
statesUq ..... 471
User Class Methods ..... 471
ctiBusyUq ..... 472
disableMedia ..... 472
enableMedia ..... 473
isMediaEnabled ..... 474
login ..... 474
loginUq ..... 475
logout ..... 475
logoutUq ..... 476
register ..... 476
setNotReady ..... 477
setPresence ..... 478
setPresenceUq ..... 478
setReady ..... 479
setWorkNotReady ..... 479
setWorkReady ..... 480
unregister ..... 481
User Class Callback Event Methods ..... 481
onCapabilitiesChanged ..... 481


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

onCtiBusy ..... 482
onCTIBUSYUq ..... 482
onCtiClear ..... 482
onDropped ..... 482
onError ..... 483
onInfo ..... 483
onLoggedIn ..... 483
onLoggedOut ..... 483
onLoggingIn ..... 484
onLoggingOut ..... 484
onMediaDisabled ..... 484
onMediaEnabled ..... 484
onNotReady ..... 485
onPresenceChanged ..... 485
onReady ..... 485
onRegistered ..... 485
onRegistering ..... 486
onSettingNotReady ..... 486
onSettingPresence ..... 486
onSettingReady ..... 486
onSettingWorkNotReady ..... 487
onSettingWorkReady ..... 487
onStat ..... 487
onStat1 ..... 487
onStat2 ..... 488
onUnknown ..... 488
onUnregistered ..... 488
onUnregistering ..... 488
onWorkNotReady ..... 489
onWorkReady ..... 489
UserCaps Class Constructor ..... 489
UserCaps ..... 489
UserCaps Class Fields ..... 490
canHandleChat ..... 490
canHandleEmail ..... 490
canHandleGeneric ..... 490
canHandleVoice ..... 491
canLogin ..... 491
canLogout ..... 491
canRefreshState ..... 491
canSetNotReady ..... 491
canSetPresence ..... 492
canSetReady ..... 492
canSetWorkNotReady ..... 492
canSetWorkReady ..... 492

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

UserData Class Constructor ..... 492
UserData ..... 493
UserData Class Fields ..... 493
UserData Class Field Constants ..... 493
UserData Class Method ..... 494
addKeyValue ..... 494
UserStatistics1 Class Constructor ..... 494
UserStatistics1 ..... 495
UserStatistics1 Class Fields ..... 495
availableCost ..... 495
ctiBusy ..... 495
currentQueue ..... 496
mostRecentTaskData ..... 496
mostRecentTaskId ..... 496
numTaskAccepted ..... 496
numTasksDone ..... 496
numTasksUnassigned ..... 497
presenceText ..... 497
reasonFlag ..... 497
state ..... 497
timeInCurrentState ..... 497
timeSinceLoggedIn ..... 498
UserStatistics2 Class Constructor ..... 498
UserStatistics2 ..... 498
UserStatistics2 Class Fields ..... 498
currentQueue ..... 499
timeIdle ..... 499
timeInCurrentState ..... 499
timeNotReady ..... 499
timeSinceLogin ..... 499
totalTimeAvailable ..... 500
totalTimeUnavailable ..... 500

**Chapter 14**

**Configuring the Email Channel ..... 501**

Understanding the Email Channel ..... 501
Handling Email ..... 502
Configuring PeopleSoft Integration Broker for the Email Channel ..... 504
Pages Used to Configure Integration Broker for Email ..... 504
Configuring the Gateway ..... 504
Configuring GETMAILTARGET Properties ..... 505
Enabling Virus Scanning ..... 512


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Locating virusscan.xml File for Your Web Server ..... 512
Configuring the virusscan.xml File ..... 513
Using Virus Scan Error Logs ..... 515
Demonstrating the Email Channel ..... 516
Pages Used to Demonstrate the Email Channel ..... 517
Using the GetMail - Server Page ..... 517
Using the MailStore - DB Page ..... 519

**Chapter 15**

**Configuring Instant Messaging in PeopleSoft MultiChannel Framework ..... 521**

Understanding Instant Messaging ..... 521
Configuring Instant Messaging ..... 522
Page Used to Configure Instant Messaging ..... 522
Configuring Instant Messaging ..... 523
Configuring XMPP Domains ..... 523
Configuring IM Users for XMPP Servers ..... 524
Viewing IMServlet Logs ..... 525
Using Single Button Presence ..... 525
Understanding Presence Detection ..... 525
Pages Used to Configure Single Presence Button ..... 526
Configuring Servers ..... 526
Configuring Screen Names ..... 527
Testing Single Presence Button ..... 527
Developing an Application with Instant Messaging Action Button for Presence Detection ..... 528
Using the Instant Messaging Sample Pages ..... 528
Pages Used to Test Instant Messaging ..... 528
Using the PeopleCode Sample Page ..... 529
Using the Button Sample Page ..... 530

**Appendix A**

**JSMCAPI Quick Reference ..... 533**

JSMCAPI Classes ..... 533
_Address ..... 533
_UQAddress ..... 534
_User ..... 535
A2AChat ..... 536
A2AChatAddress ..... 537
AgentStatistics ..... 538
AppData ..... 539
Buddy ..... 540

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Call ..... 541
CallData ..... 542
CallStatistics ..... 542
Chat ..... 543
ChatAddress ..... 544
ChatConnection ..... 546
ChatConnectionCaps ..... 548
ChatData ..... 549
Connection ..... 549
ConnectionListener ..... 550
ConnectionRequest ..... 550
Email ..... 550
EmailAddress ..... 552
EmailConnection ..... 553
EmailConnectionCaps ..... 555
EmailData ..... 555
Extension ..... 556
ExtensionCaps ..... 558
ForwardMode ..... 558
GenericAddress ..... 559
GenericConnection ..... 560
GenericConnectionCaps ..... 562
GenericData ..... 562
GenericTask ..... 563
GLOBALS ..... 564
Group ..... 565
GroupStatistics ..... 566
GroupStatistics1 ..... 567
GroupStatistics2 ..... 568
Line ..... 569
LineCaps ..... 572
MCEvent ..... 574
MediaType ..... 574
PSMC ..... 575
Reason ..... 575
RenServer ..... 576
Server ..... 577
Session ..... 578
Task ..... 580
TaskStatistics ..... 581
User ..... 582
UserCaps ..... 585
UserData ..... 585
UserStatistics1 ..... 586
UserStatistics2 ..... 587


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Appendix B**

**Installing Digital Certificates for REN SSL ..... 589**

Installing Digital Certificates ..... 589
- Installing the CA Server Certificate ..... 590
- Installing the REN Server Certificate ..... 590
- Configuring Digital Certificates ..... 591
- Importing Certificates in Java Keystore ..... 592
- Configuring the REN Server ..... 592
- Configuring REN Clusters ..... 593
- Installing Certificates for Local Node ..... 593
- Generating the Client Certificate ..... 594
- Installing PSMCAPI Certificates ..... 595

**Index ..... 597**

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# PeopleSoft MultiChannel Framework Preface

This preface introduces the PeopleSoft MultiChannel Framework.

## PeopleSoft MultiChannel Framework

PeopleSoft MultiChannel Framework delivers an integrated infrastructure to support multiple interaction channels for call center agents or other PeopleSoft users who must respond to incoming requests and notifications on these channels.

PeopleSoft MultiChannel Framework supports voice, email, web-based chat, instant messaging, and generic event channels.

## Additional Resources

The following resources are located on My Oracle Support web site:

| ***Resource*** | ***Navigation*** |
|---|---|
| Hardware and software requirements | Knowledge tab, Tools & Technology, Documentation, Hardware and Software Requirements |
| Installation guides | Knowledge tab, Tools & Technology, Documentation, Installation Guides and Notes |
| PeopleBook documentation updates | Knowledge tab, Tools & Technology, Documentation |
| Troubleshooting information | Knowledge tab, Tools & Technology, Documentation, Best Practices-Troubleshooting |

## PeopleBooks and the PeopleSoft Online Library

A companion PeopleBook called *PeopleBooks and the PeopleSoft Online Library* contains general information, including:

- Understanding the PeopleSoft online library and related documentation.
- How to send PeopleSoft documentation comments and suggestions to Oracle.
- How to access hosted PeopleBooks, downloadable HTML PeopleBooks, and downloadable PDF PeopleBooks as well as documentation updates.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Understanding PeopleBook structure.
- Typographical conventions and visual cues used in PeopleBooks.
- ISO country codes and currency codes.
- PeopleBooks that are common across multiple applications.
- Common elements used in PeopleBooks.
- Navigating the PeopleBooks interface and searching the PeopleSoft online library.
- Displaying and printing screen shots and graphics in PeopleBooks.
- How to manage the locally installed PeopleSoft online library, including web site folders.
- Understanding documentation integration and how to integrate customized documentation into the library.
- Application abbreviations found in application fields.

You can find this companion PeopleBook in your PeopleSoft online library.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 1

# Getting Started with PeopleSoft MultiChannel Framework

This chapter provides an overview of Oracle's PeopleSoft MultiChannel Framework and discusses how to implement it.

## PeopleSoft MultiChannel Framework Overview

PeopleSoft MultiChannel Framework (MCF) provides the tools that are required to support multiple channels of communication between customers (users) and agents. Some PeopleSoft applications, such as an email response management system (ERMS) from PeopleSoft CRM, use PeopleSoft MultiChannel Framework or you can develop your own applications on the framework that is provided.

PeopleSoft MultiChannel Framework delivers an integrated infrastructure to support multiple interaction channels for call center agents and other PeopleSoft users who must respond to incoming requests and notifications on these channels. PeopleSoft MultiChannel Framework supports email, web-based chat, voice, instant messaging, and generic event channels. It can be used from any PeopleSoft application.

PeopleSoft MultiChannel Framework also supports a broadcast function, which allows a user, such as a supervisor, to broadcast a notification message to a group of agents.

PeopleSoft MultiChannel Framework enables third-party routing systems that enable applications, such as PeopleSoft CRM, to provide embedded multichannel functionality. You may choose either the PeopleSoft queue server or the third-party routing server to route voice, email, chat, and generic events.

In the planning phase of your implementation, take advantage of all PeopleSoft sources of information, including the installation guides, PeopleTools documentation, and the PeopleBooks that are specific to your applications.

## PeopleSoft MultiChannel Framework Implementation

This section describes the required steps for implementing PeopleSoft MCF.

Depending on your business use of PeopleSoft MCF, several activities are necessary for implementation:

- At a minimum, you must configure a real-time event notification (REN) server.
- If you are using a PeopleSoft-supplied application, such as ERMS, you must also configure MCF servers, clusters, queues, and agents.
- You can develop your own applications built on the PeopleSoft MCF.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- If you are using PeopleSoft CTI or other third-party MCF integrations, additional configuration is required.

### *Configuring REN Servers and Clusters*

The REN server routes event notifications through the PeopleSoft MultiChannel Framework. Certain PeopleSoft applications, such as Reporting and Optimization, use the REN server to route notifications without using any of the rest of the PeopleSoft MultiChannel Framework. Therefore, a minimal configuration of PeopleSoft MultiChannel Framework includes at least one REN server and REN server cluster.

REN server-specific security setup is required, including configuration of Real-time Event Notification Permissions for each role using a REN server alone or as part of PeopleSoft MultiChannel Framework.

To provide a secure channel of communication between the clients and the REN servers, the REN servers may be SSL-enabled. SSL-enabled REN servers enable secure communication by providing client and server authentication.

Typically, a system administrator configures and manages REN servers and clusters.

See Chapter 5, "Configuring REN Servers," page 65.

### *Configuring MCF Servers, Clusters, Queues, and Tasks*

MCF servers (queue servers and log servers) work with REN servers to queue and route task notifications to agents. An implementation of PeopleSoft MultiChannel Framework requires configuration of MCF servers, MCF clusters, queues, and tasks.

Typically, a system administrator configures and manages MCF servers and clusters, and either a system administrator or an agent supervisor configures and manages queues and tasks.

See Chapter 6, "Configuring PeopleSoft MCF Servers and Clusters," page 89 and Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," page 97.

### *Creating and Defining Agents*

Agents are PeopleSoft users who are further defined as PeopleSoft MultiChannel Framework agents. Agents manage tasks that are assigned to them. Agents can log on to and accept tasks from the MultiChannel Console.

Typically, an agent supervisor defines agents and assigns agents to appropriate queues.

See Chapter 8, "Configuring PeopleSoft MCF Agents," page 119.

### *Using MCF Sample Pages*

PeopleSoft MultiChannel Framework is delivered with sample pages that demonstrate the use of email, web-based chat, generic, and instant messaging channels. An application developer can use these sample page definitions as the basis for application pages. The *PeopleTools 8.51 PeopleBook: PeopleCode API Reference* includes detailed descriptions of PeopleSoft's Mail Classes, MCFIMInfo Classes, and the Universal Queue Classes.

Additional sample pages demonstrate how to use the JavaScript MultiChannel Application Programming Interface (JSMCAPI) to customize the CTI console and to develop supervisor desktops using the available monitoring functions.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

See Chapter 9, "Administering Queues, Logs, and Tasks," page 133.

See *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Mail Classes"; *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "MCFIMInfo Class" and *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Universal Queue Classes."

### *Configuring PeopleSoft CTI*

PeopleSoft MultiChannel Framework includes support for PeopleSoft CTI. PeopleSoft CTI requires supporting computer-telephony middleware and additional configuration separate from other channels of PeopleSoft MultiChannel Framework.

See Chapter 3, "Configuring PeopleSoft Computer Telephony Integration," page 15.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

Chapter 2

# Understanding PeopleSoft MultiChannel Framework

This chapter discusses:

- PeopleSoft MultiChannel Framework.
- PeopleSoft MultiChannel Framework elements and channels.
- PeopleSoft MultiChannel Framework architecture.

## PeopleSoft MultiChannel Framework

PeopleSoft MultiChannel Framework delivers an integrated infrastructure to support multiple interaction channels for call center agents or other PeopleSoft users who must respond to incoming requests and notifications on these channels.

PeopleSoft MultiChannel Framework can be used from any PeopleSoft application.

In this context, the word *channel* refers to the technology used to communicate during an interaction. PeopleSoft MultiChannel Framework supports the following channels:

- Voice (telephone).
- Web collaboration (chat).
- Email.
- Generic tasks.
- Instant messaging.

The PeopleSoft MultiChannel Framework includes an HTML agent console, universal queueing, real-time task routing, customer-to-agent and collaborative chat, and centralized event logging.

## PeopleSoft MultiChannel Framework Elements and Channels

This section discusses:

- PeopleSoft MultiChannel Framework elements.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- PeopleSoft MultiChannel Framework channels.
- PeopleSoft MultiChannel Framework universal queue.
- PeopleSoft MultiChannel Console.

## PeopleSoft MultiChannel Framework Elements

PeopleSoft MultiChannel Framework comprises the following services and elements:

- Universal queue server, running on the Universal Queue server process (PSUQSRV).
- Real-time event notification (REN) server, running on the REN server process (PSRENSRV).
- MultiChannel Framework (MCF) log server, running on the MCFLOG server process (PSMCFLOG).
- MultiChannel console, the HTML interface through which users manage the channel interactions assigned to them.
- Chat windows, the HTML interfaces used for customer-to-agent and collaborative chat sessions.
- Agents, identified by their expertise and responsibilities.
- PeopleCode built-in functions and an email application package.
- GETMAILTARGET connector running under PeopleSoft Integration Broker.
- PeopleSoft MultiChannel Application Programming Interface (PSMCAPI) to enable server-side computer-telephony integration (CTI) integration.
- JavaScript MultiChannel Application Programming Interface (JSMCAPI) to enable a customizable CTI console and monitoring functions.

Each of these services and elements requires configuration.

In addition, each communication channel handled by PeopleSoft MultiChannel Framework requires supporting elements:

- CTI middleware to notify the system of telephone calls.
- An email server to store and serve email.
- Application pages to request customer-to-agent chat sessions and to provide context data and resolution logic for all interactions.
- Application pages or batch processes to enqueue generic events.

## PeopleSoft MultiChannel Framework Channels

This section discusses support for communications channels offered by PeopleSoft MCF.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Voice*

The agent console offers a softphone and full CTI support with Oracle-validated third-party CTI systems. Relevant application pages appear based on data attached to the call by the Interactive Voice Response (IVR) and CTI middleware.

### *Web Collaboration*

PeopleSoft application pages can include Live Help buttons that initiate customer-to-agent chat sessions. The customer and agent chat windows are browser-based and do not require a client installation or applet download. The universal queue routes chat requests to the first available agent with the skills required to handle that request. The agent chat window displays relevant customer information and enables the agent to push web content to the customer. The agent can manage multiple chat sessions from the agent console.

Agents can also include peers and supervisors in chat conferences and transfer chat sessions to other agents or queues. Agents can also initiate collaborative chats with other agents on their buddy lists.

### *Email*

PeopleSoft MCF enables applications to fetch Multipart Internet Mail Extensions emails from Post Office Protocol 3 (POP3) and Internet Message Access Protocol 4 (IMAP4) mail servers, store their parts in a database, and route the email to call center agents by either adding the email to worklists or enqueueing them on the universal queue. Large emails and binary attachments are not inserted into the database, but are instead stored in an attachment repository, accessible by URLs from a browser. The repository checks user-based and role-based security before retrieving an attachment. The email framework is built on PeopleSoft Integration Broker technology.

PeopleSoft MCF supports emails conforming to the Simple Mail Transfer Protocol (SMTP) specifications including both inbound and outbound HTML email.

### *Generic Channel*

Channels that are not provided by PeopleSoft MCF can be integrated by means of the generic channel to enqueue tasks onto the universal queue.

### *Instant Messaging*

PeopleSoft MCF enables you to use instant messaging called from an application page by using one of several instant messaging clients, such as AOL, Yahoo, or Sametime.

## PeopleSoft MultiChannel Framework Universal Queue

The universal queue accepts, evaluates, and distributes incoming task requests from multiple communication channels: email, web chat, and generic notifications.

The universal queue handles email, chat, and generic tasks. It distributes workload across the call center, or any other pool of qualified users, based on the priority of the task and the availability of agents possessing the required skill level and language skills. Availability is based on agent presence and the cost of the new task (a measure of the task's impact on agent capacity) against the current workload of each agent.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Agents can forward tasks to other agents or to another queue. The task is removed from the transferring agent's workload and added to the accepting agent's workload.

Email and generic tasks that are not closed before the agent signs out persist in the database. Persisted tasks are reassigned to the same agent that accepted the tasks when the agent signs in again. A task that is not accepted, within configurable time limits, by the agent to whom it was assigned is reassigned to another qualified agent, if one is available. Tasks that are not resolved within configurable time limits are automatically escalated. Tasks that cannot be assigned to or are not accepted by any agent within configurable time limits are moved to an overflow table.

Voice tasks (CTI) are not queued or routed by the universal queue. They take precedence over all other tasks. However, the queue server adds the cost of voice tasks to the agent workload calculations it uses to queue and assign incoming tasks.

### PeopleSoft MultiChannel Console

The agent console is the web-browser-based desktop from which the user manages all tasks, irrespective of channel. The console combines CTI, chat, email, and generic notice response tools into one configurable window. Agents use the console to sign in, to select their current queue, to accept tasks, and to initiate and accept collaborative chat requests with buddy users. After the agent accepts a task, additional browser-based windows appear to enable the agent's response. These windows are task-dependent and include elements developed specifically for supporting applications, such as email response management systems.

A custom CTI console can be created by means of the JSMCAPI.

## PeopleSoft MCF Architecture

This section discusses:

- PeopleSoft MCF server architecture.
- Chat architecture.
- Email architecture.

### PeopleSoft MCF Server Architecture

The REN server (the PSRENSRV process) is essential to the framework architecture. MCF events are sent to REN servers, which then deliver them to recipients of those topics. The REN server is a modified web server using the HTTP 1.0 or HTTP 1.1 communications protocol. Communication with server processes and MCF browser windows is bidirectional, because the browser windows maintain persistent connections to the REN server. Events can be sent proactively to browser windows without polling or page refreshes. To provide a secure channel of communication to overcome concerns from PeopleSoft customers about sensitive data and its security, the REN server can be Secure Sockets Layer (SSL)-enabled. An SSL-enabled REN server provides secure communication that encrypts and provides client and server authentication.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Applications send interaction and action requests (tasks) received from the supported communication channels to logical queues. The REN server notifies the universal queue server (the PSUQSRV process) responsible for that queue that a new task has arrived. Tasks are queued in order of priority until they can be assigned to an available agent qualified to respond to the task, at which time the queue server sends an assignment notification to the user's MultiChannel Console through the REN server.

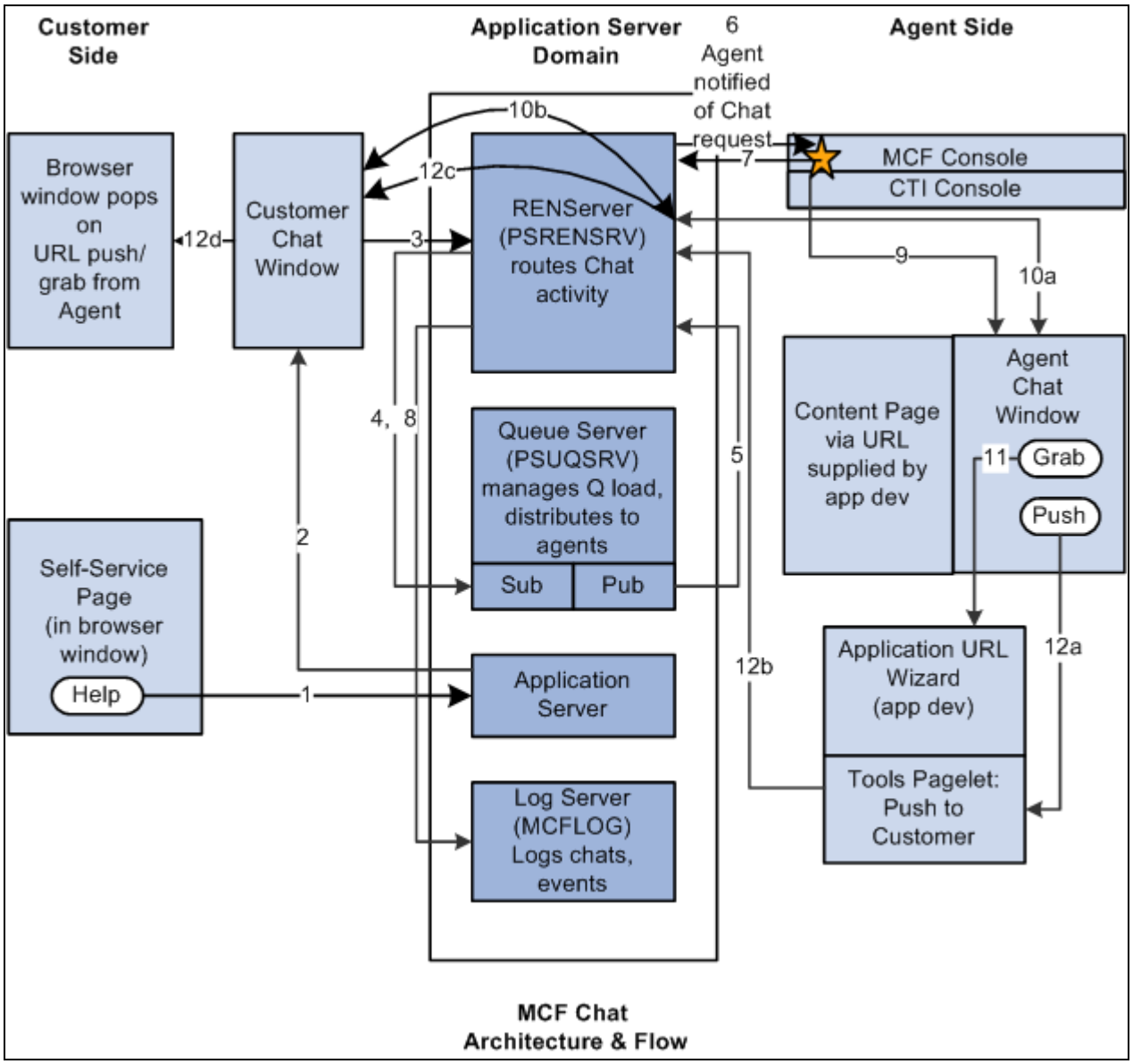
The queue server routes work requests (tasks) to users based upon a set of configurable policy properties that define which agents can handle what types of tasks and when the tasks can be assigned. The queue server manages state information about the current status of active agents and active tasks.

The MCF log server logs MCF events and chat content to the database. You configure logging levels on the MCF administration pages.

## Chat Architecture

This diagram illustrates the architecture of a chat session:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




MCF chat architecture and flow

When a customer clicks the Help button on an application page:

1. InitChat() passes the following parameters to the application server:

   - Queue number.
   - Priority override.
   - Context page URL.
   - Query.

     See *PeopleTools 8.51 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions."


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

2. The application server posts to the REN server to notify the queue server that a customer chat is waiting.

   The application server then returns the name and port of the REN server and the iScript to build the customer chat window.

3. The customer chat window appears and communicates with the REN server to receive all events on that chat topic.

4. The REN server notifies the queue server that a customer chat is waiting.

   The queue server determines the appropriate agent according to workload, cost, agent availability, skill level, and language.

5. The queue server tells the REN server to notify the agent that the agent has been assigned a chat.

6. The REN server notifies the agent from the MultiChannel Console that the agent has been assigned a chat.

7. The agent, from the MultiChannel Console, notifies the REN server to notify the queue server that the agent has accepted the task.

8. The REN server notifies the queue that the agent has accepted the task.

9. The MultiChannel Console displays an agent chat window.

   The agent responds to inquiry.

10. Agent to customer two-way communication is mediated by the REN server.

11. If the agent chooses to grab a URL, the Grab button invokes the application URL wizard.

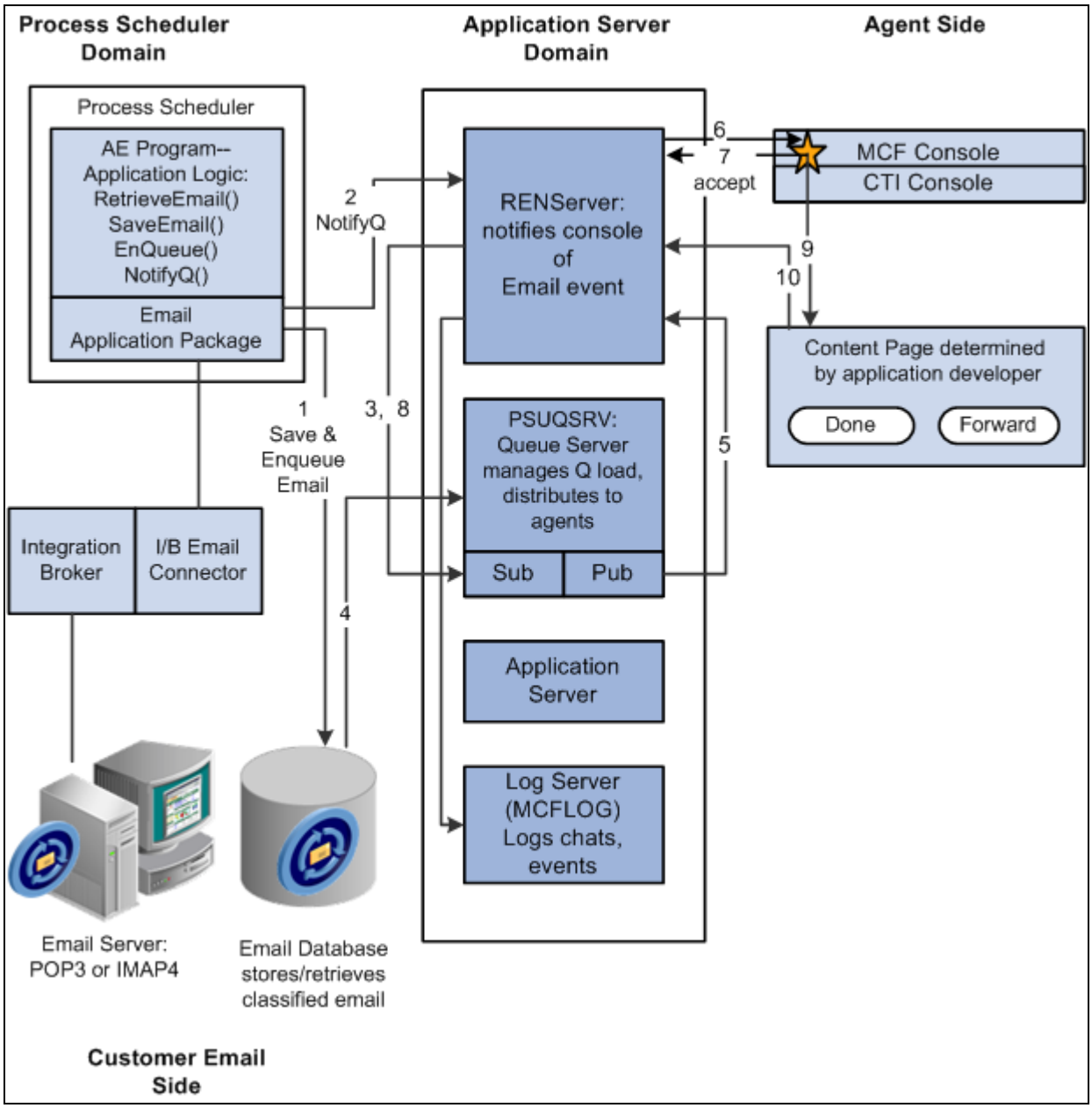
12. If the agent chooses to push a URL, the Push button invokes a pagelet (labeled with an *A* in the diagram) to push a selected URL through the REN server (labeled with a *B* in the diagram) and customer chat window (labeled with a *C* in the diagram) to a new browser window.

The entire chat session can be logged through the MCF log server (labeled with a dotted line in the preceding diagram).

## Email Architecture

This diagram illustrates the architecture of email processing:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




MCF email architecture and flow

When a customer sends an email:

1. A PeopleSoft Application Engine program uses the MCF email application package classes and PeopleCode built-in functions to save and enqueue email in a database.
2. The PeopleSoft Application Engine program notifies the REN server that email has been enqueued.
3. The REN server notifies the queue server of the waiting email.
4. The queue server retrieves email information required to determine appropriate routing from the database.

   The queue server determines the appropriate agent to handle each email according to workload, cost, agent availability, skill level, and language.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

5. The queue server tells the REN server to notify the agent that the agent has been assigned an email.
6. The REN server notifies the agent from the MultiChannel Console that the agent has been assigned an email.
7. The agent, from the MultiChannel Console, notifies the REN server to notify the queue server that the agent has accepted the task.
8. The REN server notifies the queue that the agent has accepted the task.
9. The MultiChannel Console displays an agent email window, as determined by the application developer.

   The agent responds to the email.
10. The agent's resolution of the task is communicated back to the REN server by either the Done or Forward button.

Email events can be logged to a database by the MCF log server.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 3

# Configuring PeopleSoft Computer Telephony Integration

This chapter provides overviews of the PeopleSoft Computer Telephony Integration (CTI) console, adapter-based CTI requirements, and applet-based CTI requirements and discusses how to:

- Configure PeopleSoft CTI.
- Configure PeopleSoft CTI using adapters.
- Configure PeopleSoft CTI using applets.
- Configure PeopleSoft CTI queues and CTI agents.
- Use other PeopleSoft CTI options.
- Use the PeopleSoft CTI sample pages.

## Understanding the PeopleSoft CTI Console

This section discusses:

- PeopleSoft CTI.
- PeopleSoft CTI components.

### PeopleSoft CTI

PeopleSoft CTI enables you to integrate your PeopleSoft applications with your call center. PeopleSoft CTI offers the following benefits:

- Seamlessly integrates your PeopleSoft application with Oracle-validated third-party CTI systems to improve agent productivity.

  *Agents* refers to the individuals who interact with your customers using CTI.

- Requires only that you install a supported web browser on the agent's workstation.
- Enables agents to take advantage of browser-based call management and automatic population of PeopleSoft transaction pages with the relevant customer data associated with an incoming call.
- Transmits DTMF data.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Handles outbound calls from automated systems.

PeopleSoft CTI is an optional component that you can integrate with the PeopleSoft MultiChannel Framework. This means that you can incorporate a CTI channel within the MultiChannel Framework. PeopleSoft CTI requires third-party middleware in the form of an Oracle-validated third-party CTI middleware.

---

**Note.** For a list of partners that offer CTI middleware integrations, refer to the link in the See Also section.

---

In PeopleSoft CTI, the CTI middleware performs the call routing. The universal queue is not involved in routing calls. For an incoming call, the CTI middleware notifies the MultiChannel Console, which then notifies the queue server so that the agent's workload can be updated with the cost of a call.

### *See Also*

For vendors using the adpater-based CTI solution, refer to Oracle Validated Application Integrations – Find a Partner Solution http://www.oracle.com/partnerships/isv/integration/search.html

For vendors using the applet-based CTI solution, sign in to My Oracle Support and select Product Certifications tab.

## PeopleSoft CTI Components

The PeopleSoft CTI Console works together with the IVR (Interactive Voice Response) system, the CTI middleware, an Automatic Call Distributor (ACD), and your PeopleSoft application.

When a customer calls, the caller enters his or her information (for example, an account number) using the IVR system. Using this information, the CTI middleware routes the call to an appropriate ACD queue. The ACD sends the call to the next available agent on that queue and notifies the CTI middleware that an incoming call is on that Directory Number (DN). The CTI middleware, in turn, notifies the CTI Console and passes the customer's information as attached data. The CTI Console uses the attached data to determine what PeopleSoft transaction page to open (pop-up) for the agent and what application data to retrieve from the database. The agent can manage the call using the CTI Console, which in turn communicates with the PBX (Private Branch Exchange) using the CTI middleware.

You can configure PeopleSoft CTI systems using either the adapter solution or the applet solution. These terms are defined in the following table.

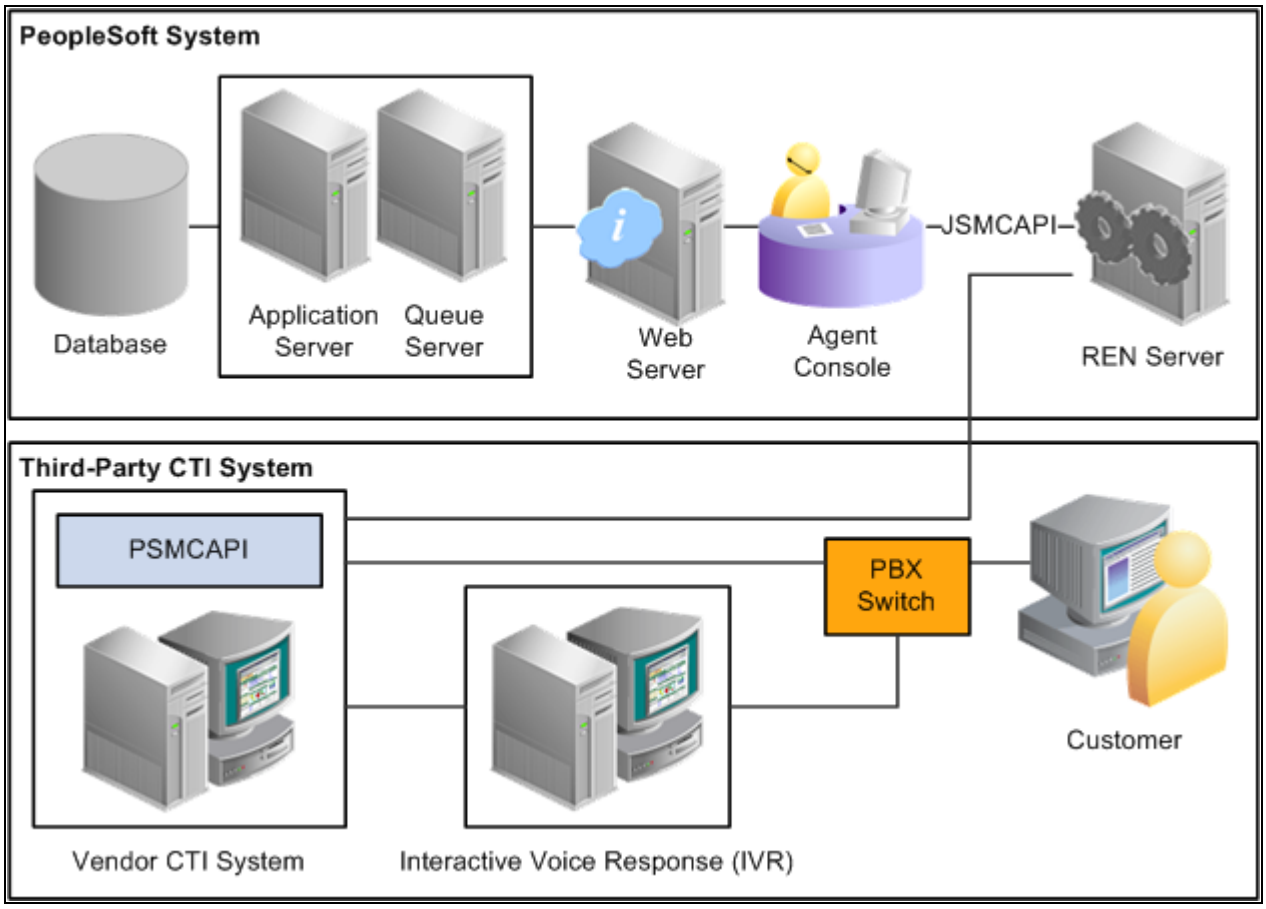
| | |
|---|---|
| **Adapter** | For PeopleSoft CTI systems using PeopleTools 8.45.05 or later, the CTI Console uses a JavaScript MultiChannel Application Programming Interface (JSMCAPI) to communicate with CTI middleware that implements the PeopleSoft MultiChannel Application Programming Interface (PSMCAPI). |
| **Applet** | The CTI Console uses a Java applet that runs within the web browser to communicate directly with your CTI middleware. The CTI Console communicates to the Java applet using JavaScript. The applet is delivered in a file called pCti.cab, which is approximately 500 KB. The applet resides in the browser cache to reduce network traffic and improve response time. |

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Note.** The CTI Console Java applet was deprecated in PeopleTools 8.46. Deprecated features are supported; however, no additional functionality is planned. Refer to the Product Certifications link on My Oracle Support, https://metalink3.oracle.com (sign-in required), for more information about supported products and deprecated features.

Configure PeopleSoft CTI to use either the applet or the non-applet (JSMCAPI) console on the CTI Type tab on the Configure CTI page.

This diagram illustrates PeopleSoft CTI architecture using PSMCAPI and JSMCAPI:




PeopleSoft CTI integration architecture

### *See Also*

*PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Internet Script Classes (iScript)"

## Adapter-Based CTI Requirements

This section describes components required for PeopleSoft CTI using the adapter-based solution. It discusses:

- PeopleSoft MultiChannel API (application program interface)

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Required Security for PSMCAPI
- JavaScript MultiChannel API

## PeopleSoft MultiChannel API

The PeopleSoft MultiChannel API (PSMCAPI) is a Java API and software development kit (SDK) that provides server-side connectivity with the PeopleSoft CTI system. PSMCAPI enables third-party telephony vendors and system integrators to integrate with PeopleSoft applications.

PSMCAPI and JSMCAPI are installed during PeopleTools installation.

This table provides the installed locations of PSMCAPI and JSMCAPI components:

| ***Component*** | ***Location*** |
|---|---|
| PSMCAPI Java archive | <PS_HOME>\sdk\psmcapi\dist\lib |
| PSMCAPI configuration files:<br>• psmcapilog.properties<br>• renclient.properties | <PS_HOME>\sdk\psmcapi\dist\config |
| JSMCAPI JavaScript file | <PIA_HOME>\webserv\<domain>\applications\peoplesoft\PORTAL\ps\pMCF |

### *Working With Older Adapters*

For some implementations using older versions of third party adapters, you may require a version of psmcapi.jar built with an older version of JDK. PeopleTools provides a previous version of the PSMCAPI for use with older adapters.

This table provides the installed locations of the PSMCAPI components provided for use with older adapters:

| ***Component*** | ***Location*** |
|---|---|
| PSMCAPI Java Archive | PS_HOME\sdk\psmcapi\dist\archive\lib |
| PSMCAPI configuration files | PS_HOME\sdk\psmcapi\dist\archive\config |

### *Configuring PSMCAPI*

Two files, psmcapilog.properties and renclient.properties, include parameters to configure PSMCAPI. Configure the logging characteristics of PSMCAPI in psmcapi.properties. To have PSMCAPI generate full debug logs, set com.peoplesoft.pt.mcf.level to FINEST.

The following table lists all the parameters in renclient.properties:


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Parameter* | *Default Value* | *Description* |
|---|---|---|
| interval_heartbeat_server | 30000 | Heartbeat interval in milliseconds from PSMCAPI to console. |
| interval_heartbeat_client | 30000 | Heartbeat interval in milliseconds from console to PSMCAPI. |
| maxsize_eventqueue | 100000 | Maximum number of events in the event queue. |
| interval_event_expired | 300000 | Expiration interval in milliseconds for an event in the event queue. An expired event is never published to the client/console and is discarded from the event queue. |
| interval_topic_reaper | 3600000 | Topic reaper interval in milliseconds. |
| number_of_requests_in_requestschunk | 1 | The number of requests that JSMCAPI sends at a time during auto-recovery. |
| waitingtime_between_requestchunks | 1 | The time elapsed between two consecutive JSMCAPI request queues during auto-recovery. |
| mtu_size | 0 | The Maximum Transmission Unit size of your computer or network. Set mtu_size to 0 (zero) for no TCP packet padding, or to the maximum transmission unit (MTU) size for your network or computer to remove TCP acknowledgement delays. |
| heartbeats_to_miss | 2 | The number of heartbeat intervals to wait before removing a nonresponsive client |
| psmcapi_heartbeats_to_miss | 5 | The number of PSMCAPI heartbeat intervals to wait before removing a nonresponsive client. |
| tcp_nodelay | True | TCP no delay. Set to True to disable the TCP Nagle algorithm. |
| disable_session | False | This parameter is used when multiple PSMCAPI implementations subscribe to the same REN server.<br>Set to True to disable the internal session and heartbeat listeners. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Required Security for PSMCAPI

CTI agents require only that the MCF Agent security object be enabled on the REN Permissions page of their associated permission list. The MCF Agent security object includes security for both CTI and other MCF channels.

To enable CTI configuration pages, authorize the CTI Type, CTI Applet, and CTI Non-Applet panel items on the PT_CTI page permissions for the appropriate permission list.

Also enable WEBLIB_MCF weblib permissions for the appropriate permission list.

***See Also***

Chapter 5, "Configuring REN Servers," Configuring REN Servers, page 71

*PeopleTools 8.51 PeopleBook: Security Administration*, "Getting Started with Security Administration"

## JavaScript MultiChannel API

The JavaScript MultiChannel Application Programming Interface (JSMCAPI) is an interface that application developers can use to generate the CTI Console or to enable CTI functionality on a PeopleSoft Pure Internet Architecture page. The JSMCAPI builds on the real-time event notification (REN) JavaScript client. JSMCAPI uses standard JavaScript.

***See Also***

Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using and Demonstrating JSMCAPI, page 178

# Applet-Based CTI Requirements

PeopleSoft CTI using the adapter based solution uses:

- Java Applet Console.
- Java Runtime Environment (JRE).

## Java Applet Console

The Java applet console was deprecated in PeopleTools 8.46. Specific Java APIs are shipped as a part of the CTI applet. Depending on the CTI software and version, your PeopleSoft application may require additional CTI components to attach data to incoming calls.

This section lists some considerations for:

- Genesys components

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Cisco components

### *Genesys Components*

If you are using a Genesys CTI system with the PeopleSoft CTI Java applet console, the assumption is that you already have a functioning Genesys system configured at your site. Genesys Java API is shipped as part of the CTI applet; however, no other Genesys products are shipped with the PeopleSoft system.

To interact with the PeopleSoft CTI system, you need a Genesys T-Server installed and configured before you begin installing your PeopleSoft CTI system. Refer to your Genesys documentation for installation information.

In addition, your PeopleSoft application may require the following CTI components to attach data to incoming calls:

- Genesys Strategy Builder or Interaction Router or equivalent.
- An IVR supported by Genesys and capable of passing call data to Genesys.

For detailed information regarding specific versions that Oracle supports, refer to the Product Certifications on the My Oracle Support web-site.

### *Cisco Components*

If you are using a Cisco CTI system with the PeopleSoft CTI Java applet console, the assumption is that you already have a functioning and configured Cisco system at your site. The Cisco Java API is shipped as part of the CTI applet; however, no other Cisco products are shipped with the PeopleSoft system.

You need a Cisco ICM Central Controller Server installed and configured before you begin installing your PeopleSoft CTI system.

In addition, your PeopleSoft application may require an IVR supported by Cisco and capable of passing call data to Cisco. This enables CTI components to attach data to incoming calls.

---

**Note.** For detailed information regarding specific versions that Oracle supports, refer to the Product Certifications on the My Oracle Support web-site.

---

## Java Runtime Environment

Java applets, such as the PeopleSoft CTI Java applet console, run within a Java Virtual Machine (JVM) that uses a *sandbox* (a controlled environment in which remote programs run) to restrict access to what the applets can do. The JRE is required for those PeopleSoft CTI systems using the Java applet console. The JRE is not required if you are using the JSMCAPI console.

If you are using the Java applet console, you should install and use the Sun Java Plug-in 1.4.2 or later. You can download the plug-in from Sun Microsystems at www.java.com.

### *Applet Protection*

The CTI applet has an attached digital signature, which enables the PeopleSoft CTI applet to open a network connection to your CTI system as opposed to being able to send back data only to the web server that downloaded it.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

The digital signature protects the applet against tampering, and it requires that the agent grant permission to run the applet. When doing so, agents need to indicate whether such permission should be granted to any PeopleSoft code for all subsequent sessions or for the current session only. Instruct your end users to select the appropriate option for your site.

### *See Also*

Supported web browsers for PeopleSoft PeopleTools Certifications on the My Oracle Support web-site.

# Configuring PeopleSoft CTI

This section describes how to install and configure the PeopleSoft CTI system. This information assumes that you already have a functioning CTI system installed and configured at your site. The information in this section is common to both the adapter and applet solutions.

This section discusses how to:

- Install PeopleSoft CTI.
- Enable PeopleSoft CTI.
- Create CTI configurations.
- Use the Shared Phone Book page.
- Use the Miscellaneous page.
- Use the Reason Code page.

## Pages Used to Configure CTI

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| CTI Type | CTICONSOLETYPE | • PeopleTools, MultiChannel Framework, CTI Configuration, CTI, CTI Type<br>• PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, CTI Type | Configure CTI console type. CTI console can be either Java applet or CTI JavaScript (non-applet) console. |

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Shared Phone Book | PT_CTI_PNCTI | • PeopleTools, MultiChannel Framework, CTI Configuration, CTI, Shared Phone Book<br>• PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, Shared Phone Book | Create a list of frequently dialed phone numbers for use with a specified configuration. |
| Miscellaneous | PT_CTI_MISC | PeopleTools, MultiChannel Framework, CTI Configuration, Miscellaneous | Enter the default screen pop-up window URL.<br>**Note.** This page is available for the CTI server only. |
| Reason Code | CTI_SYSREASON | • PeopleTools, MultiChannel Framework, CTI Configuration, CTI, Reason Code<br>• PeopleTools, MultiChannel Framework, Third-Party Configuration, ,CTI Configuration, Reason Code | Define your own customized codes when using a third-party routing system.<br>**Note.** This page is available for third-party routing systems only. |

## Installing PeopleSoft CTI

When you run the PeopleSoft Pure Internet Architecture setup program, the PeopleSoft CTI files are installed automatically to your web server.

---

**Note.** You do not need to select any additional options from the install program dialogs. The CTI files are installed by default.

---

After you have run the PeopleSoft Pure Internet Architecture setup program, enable the PeopleSoft CTI Console and configure the system as discussed in the following sections.

### *See Also*

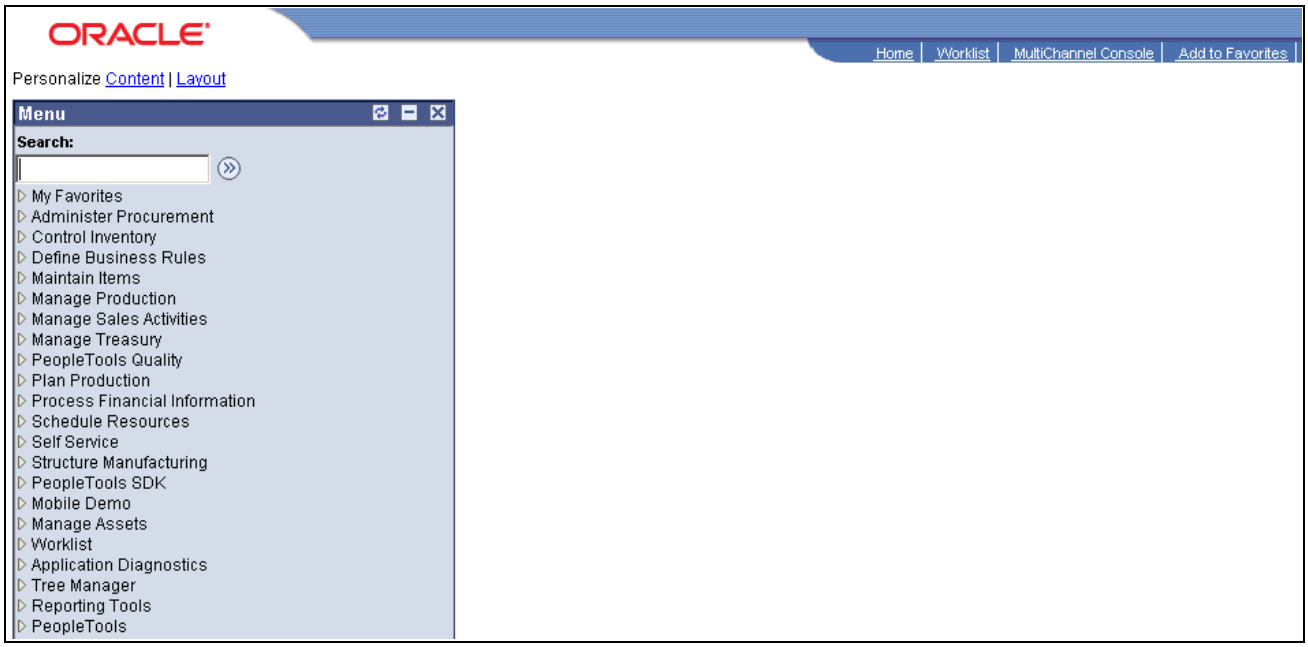
PeopleTools Installation Guide for your platform

CTI vendor product documentation

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Enabling PeopleSoft CTI

If a user is set up as a CTI agent or a MultiChannel Framework agent, the Multichannel Console link appears in the universal navigation header (in the upper-right portion of the screen) when he or she accesses the PeopleSoft Pure Internet Architecture. The following example shows the Multichannel Console link displayed in the PeopleSoft Pure Internet Architecture:




The MultiChannel Console link displayed in the PeopleSoft Pure Internet Architecture

Users who do not have full access to the MCF Agent object do not see this link.

---

**Note.** Blended agents—agents who use both the CTI and MCF consoles—must sign in to queues supported by the same REN server cluster.

---

See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the CTI Sample Console, page 184.

### *See Also*

*PeopleTools 8.51 PeopleBook: PeopleTools Portal Technologies*, "Understanding Portal Technology"

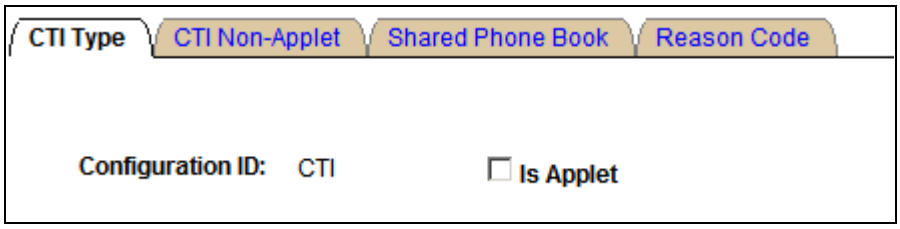
Chapter 3, "Configuring PeopleSoft Computer Telephony Integration," Configuring CTI Agents, page 38

## Configuring CTI Console Type

Access the CTI Type page using either of the following navigation paths, whichever is appropriate to you:


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

PeopleTools, MultiChannel Framework, CTI, CTI Type (if you are using the CTI server) or PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, CTI Type (if you are using a third-party routing server).




CTI Type page

Select the Is Applet check box if you are using the CTI Java applet console.

Clear the Is Applet check box if you are using the CTI JavaScript (JSMCAPI) console.

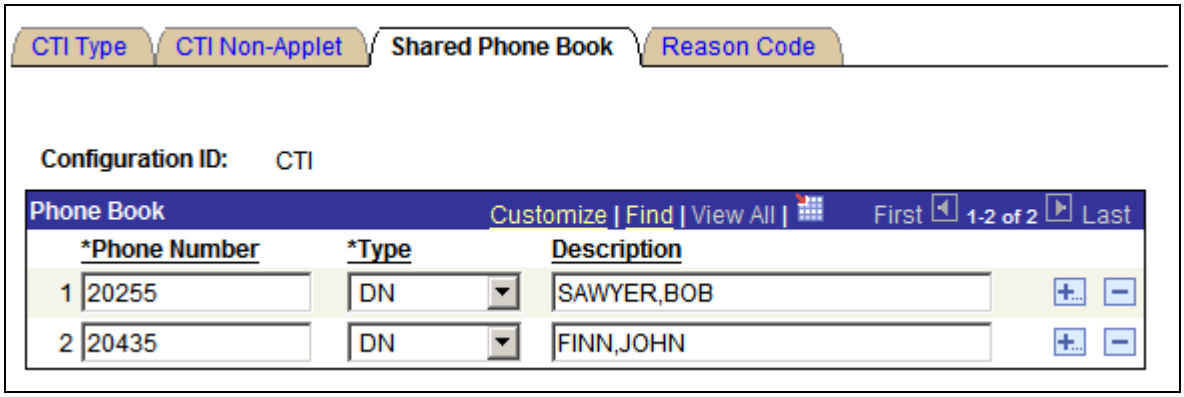
A CTI configuration contains all the information required for a user to be able to connect to a CTI server.

**Note.** The Configuration ID is created when CTI is configured for the first time.

## Creating a List of Frequently Dialed Phone Numbers

Access the Shared Phone Book page using either of the following navigation paths, whichever is appropriate to you:

PeopleTools, MultiChannel Framework, CTI, Shared Phone Book (if you are using the CTI server) or PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, Shared Phone Book (if you are using a third-party routing server).




Shared Phone Book page

On this page, you can manage a list of frequently dialed phone numbers for a specific CTI configuration. These numbers appear when an agent connected to that CTI configuration selects the drop-down list box when dialing a number from the CTI Console. This saves the agents from manually entering any dialed numbers present in that CTI configuration when making outbound calls.

**Note.** Phone lists are updated on the CTI Console only after the CTI Console launches. To refresh phone lists, refresh the browser and reactivate the console.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Phone Number** Enter a frequently dialed phone number to associate with this configuration.

**Type** Select one of the following values:

- *DN* (directory number): The number that identifies a telephone set on a PBX or in the public network. The caller dials this number to establish a connection to the addressed party. The DN can be a local PBX extension (a local DN) or a public network telephone number.
- *Queue* The directory number that identifies an ACD queue or group. Calls to a group are distributed to agents belonging to the group, according to ACD algorithms.

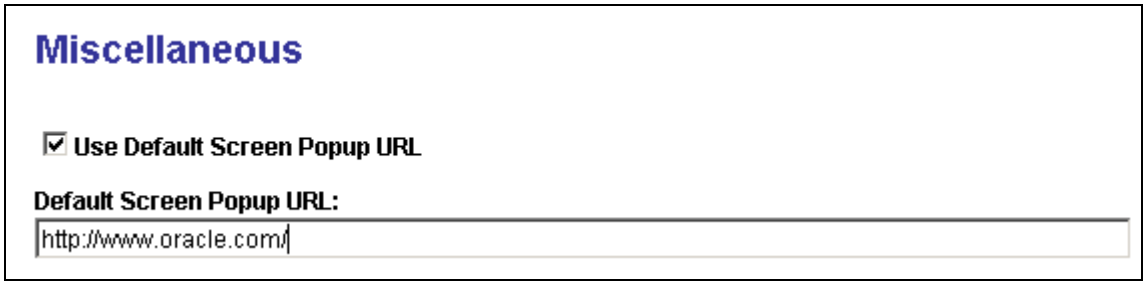
**Description** Add a description for the telephone number.

## Entering Default Screen Pop-Up URL

Access the Miscellaneous page using the following navigation path:

PeopleTools, MultiChannel Framework, CTI, Miscellaneous

**Note.** This page is available through the CTI server only.




The Miscellaneous page having the Use Default Screen Popup URL check box and the Default Screen Popup URL editable field

Set default screen pop-up URL parameters on the Miscellaneous page.

**Use Default Screen Popup URL** Select if you want to set a default URL for the pop-up screen.

The system uses the default method to determine what URL to use for the page launched for incoming calls. The system uses that default URL to examine the user data attached to that call. However, for customers who do not want to attach this user data to incoming calls, this option enables you to use the same URL for all pop-up screens.

After selecting this option, enter the URL in the Default Screen Popup URL edit box.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Default Screen Popup URL**

If the Use Default Screen Popup URL check box is selected, enter the value for the default URL.

**Note.** To ensure that the user does not have to sign in to the pop-up window, the domain of the default URL must exactly match the domain of the sign-in URL. If the sign-in URL has no domain, the machine name of the default URL should be the same as the machine name of the sign-in URL. If either the domain or the machine name do not match, the system prompts the user for the user ID and password for the first pop-up screen.

With Matching Domains:

Within the same domain, such as example.com, the machine names can be different. For example:

*Signon URL:*

http://ntserver1.example.com/peoplesoft8/signon.html

*Default URL:*

http://uxserver2.example.com/servlets/iclientservlet/peoplesoft8/?ICType=PANEL&Menu=UTILITIES&Market=GBL&Component=MESSAGE_CATALOG1&Target=Main1&LANGUAGE_CD=ENG&MESSAGE_SET_NBR=1

Without Matching Domains:

Without the matching domains, the machine name should be the same in both URLs. For example:

*Signon URL:*

http://ntserver1/peoplesoft8/signon.html

*Default URL:*

http://ntserver1/servlets/iclientservlet/peoplesoft8/?ICType=PANEL&Menu=UTILITIES&Market=GBL&Component=MESSAGE_CATALOG1&Target=Main1&LANGUAGE_CD=ENG&MESSAGE_SET_NBR=1

### *See Also*

*PeopleTools 8.51 PeopleBook: Security Administration*, "Implementing Single Signon"

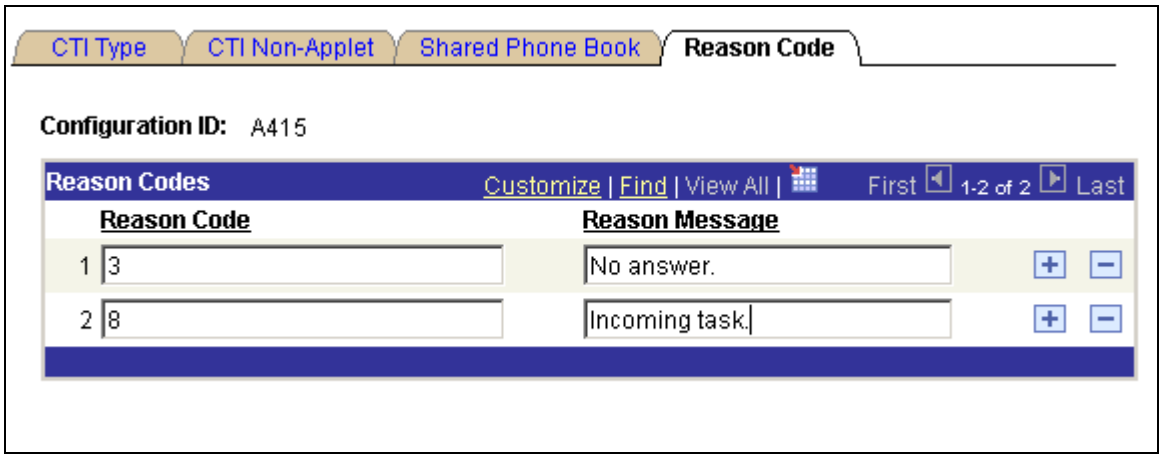
## Using the Reason Code Page

Access the Reason Code page using the following navigation path:

PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, Reason Code

**Note.** This page is available only to third-party vendors.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Reason Code page having the following editable fields: Reason Code and Reason Message

This page is used by the third-party vendors to define their customized codes.

***See Also***

Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Understanding JSMCAPI, page 178

# Configuring PeopleSoft CTI Using Adapters

This section discusses how to configure the CTI non-applet console.

## Page Used to Configure the CTI Non-Applet Console

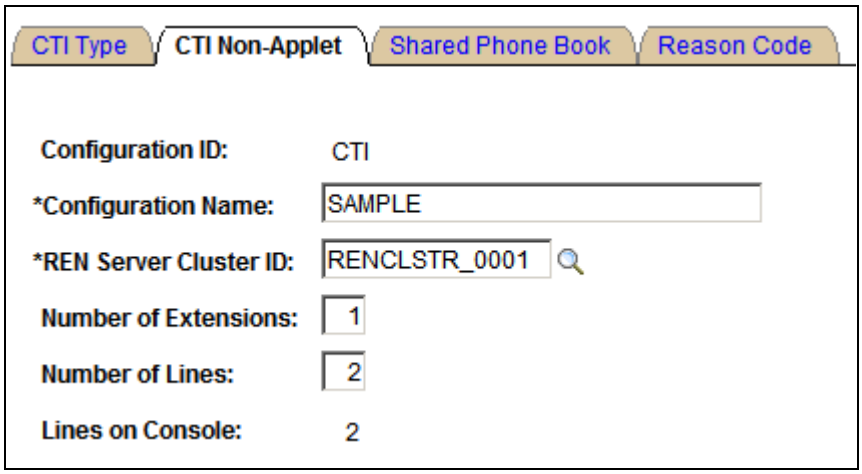
| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| CTI Non-Applet | CTIREN | • PeopleTools, MultiChannel Framework, CTI Configuration, CTI, CTI Non-Applet<br>• PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, CTI Non-Applet | Configure CTI for use with the JavaScript console. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Configuring the CTI Non-Applet Console

Access the CTI Non-Applet page using either of the following navigation paths, whichever is appropriate to you, to configure the CTI non-applet (JSMCAPI) console:

PeopleTools, MultiChannel Framework, CTI, CTI Non-Applet (if you are using the CTI server) or PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, CTI Non-Applet (if you are using a third-party routing server).




CTI Non-Applet page

**Configuration ID** Displays the name of the CTI configuration. The name cannot be modified after it is created.

**Configuration Name** Add a descriptive name to help identify the configuration.

**Number of Extensions** Enter the number of extensions or directory numbers associated with the telephone.

**Number of Lines** Enter the number of lines associated with each extension. Depending on your configuration, you can enter up to two lines.

**Lines on Console** The CTI Console supports up to two lines. The only supported configurations are two extensions with one line each, one extension with two lines, and one extension with one line.

# Configuring PeopleSoft CTI Using Applets

This section discusses how to:

- Configure the CTI applet console
- Use the CTI Genesys page

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Use the CTI Cisco page

## Pages Used to Configure CTI Using Applets

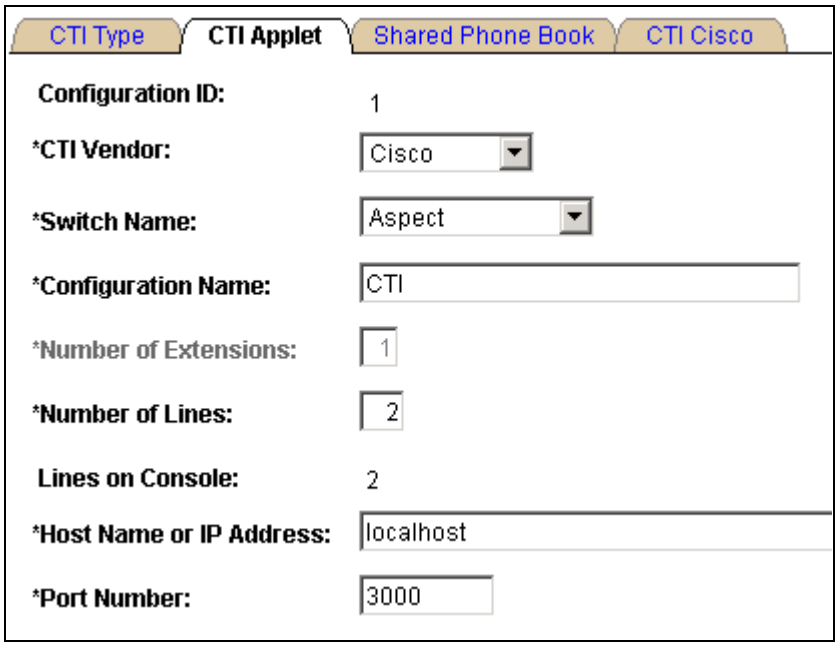
| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| CTI Applet | PT_CTI_CONFIG | • PeopleTools, MultiChannel Framework, CTI Configuration, CTI, CTI Applet<br>• PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, CTI Applet | Configure CTI for use with the Java applet console. |
| CTI Cisco | PT_CTI_CSCO | • PeopleTools, MultiChannel Framework, CTI Configuration, CTI, CTI Cisco<br>• PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, CTI Genesys | Configure parameters specific to Cisco CTI middleware for use with the Java applet console. |
| CTI Genesys | PT_CTI_GENESYS | • PeopleTools, MultiChannel Framework, CTI Configuration, CTI, CTI Genesys<br>• PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, CTI Genesys | Configure parameters specific to Genesys CTI middleware for use with the Java applet console. |

## Configuring the CTI Applet Console

Access the CTI Applet page using either of the following navigation paths, whichever is appropriate to you, to configure PeopleSoft CTI for use with the CTI applet console.

PeopleTools, MultiChannel Framework, CTI, CTI Applet or PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, CTI Applet.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The CTI Applet page displaying the Configuration ID and having the following editable fields: CTI Vendor, Switch Name, Configuration Name, Number of Extensions, Number of Lines, Lines on Console, Host Name or IP Address, and the Port Number.

**Configuration ID** Displays the name that you gave the configuration when you created it. The name cannot be modified after it is created.

**CTI Vendor** Select the vendor of your CTI solution. The options are:

- *Genesys*
- *Cisco*

This value controls whether the CTI Cisco page or the CTI Genesys page appears in the component. For example, if you select *Cisco,* the CTI Cisco page appears.

**Switch Name** Select from one of the supported switches.

**Configuration Name** Add a descriptive name to help identify the configuration.

**Number of Extensions** Enter the number of extensions or directory numbers associated with the telephone.

For Genesys, *1* or *2* can be used.

For Cisco, only *1* can be used.

**Number of Lines** Enter the number of lines associated with each extension. Depending on your configuration, you can specify up to two lines.

If Number of Extensions is *1*, enter *2*.

If Number of Extensions is *2*, enter *1*.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Lines on Console** The CTI Console supports up to two lines. The only supported configurations are two extensions with one line each, one extension with two lines, and one extension with one line.

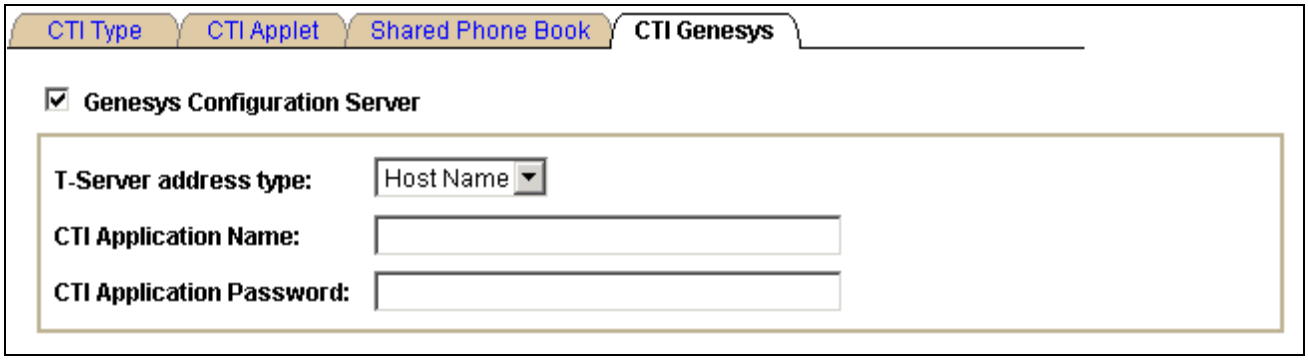
**Host Name or IP Address** Enter the host name or IP address for your CTI server.

**Port Number** Enter the port number on which the T-Server, Configuration Server, or Cisco ICM listens.

## Using the CTI Genesys Page

Access the CTI Genesys page using either of the following navigation paths, whichever is appropriate to you:

PeopleTools, MultiChannel Framework, CTI, CTI Genesys or PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, CTI Genesys.




The CTI Genesys page having the Genesys Configuration Server check box, T-Server Address Type drop-down list, CTI Application Name, and CTI Application Password

The CTI Genesys page provides additional options for Genesys implementations. This page appears only when Genesys is selected as the CTI vendor on the General Configuration page and the chosen CTI type is Applet.

**Genesys Configuration Server** Select to direct the CTI Console to get data required for connecting to the T-Server from a configuration server instead of from the PeopleSoft database. Using a configuration server is transparent to the user or agent. When the agent activates the console, it requests a list of available T-Servers from the Configuration Server, and then the console sequentially attempts to connect to each T-Server in that list until it establishes a connection. If it reaches the end of the list without connecting to a T-Server, an error message is returned to the agent.

**Note.** If you select this option, you enable the Application User Information group box on this page. Enter the appropriate application user name and password for the Genesys system.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**T-Server address type**

If you are using a configuration server to indicate what T-Server to connect to (instead of connecting directly to a T-Server), indicate whether the configuration server is returning the host name or the IP address of the T-server.

Enter the type address to use when sending information to the T-Server. The options are *Host Name* and *IP Address.* Your selection should correspond to the host name or IP address that you entered on the CTI Applet page.

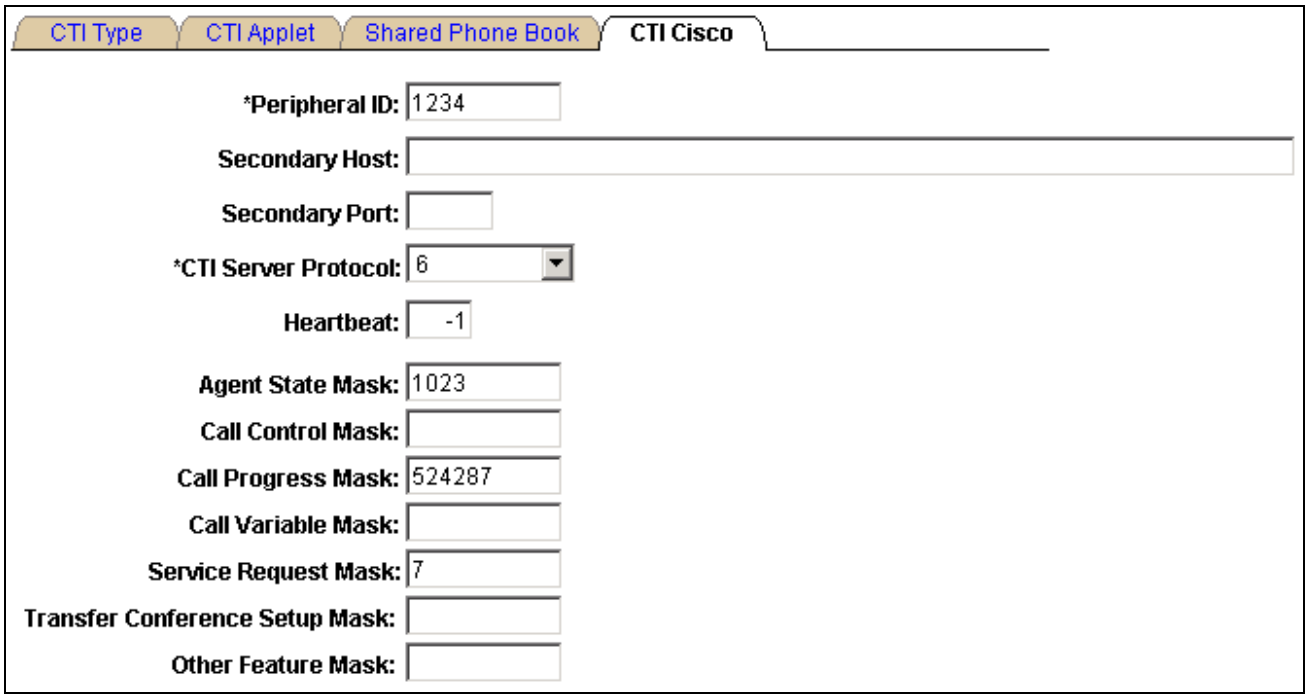
**CTI Application Name** and **CTI Application Password**

Enter the Genesys application name and password used to sign in to the Genesys Configuration Server. Using this option, each user connects to the configuration server with the same application name.

## Using the CTI Cisco Page

Access the CTI Cisco page using either of the following navigation paths, whichever is appropriate to you:

PeopleTools, MultiChannel Framework, CTI, CTI Cisco or PeopleTools, MultiChannel Framework, Third-Party Configuration, CTI Configuration, CTI Cisco.




The CTI Cisco page having the following editable fields: Peripheral ID, Secondary Host, Secondary Port, CTI Server Protocol, Heartbeat, Agent State Mask, Call Control Mask, Call Progress Mask, Call Variable Mask, Service Request Mask, Transfer Conference Setup Mask, and the Other Feature Mask.

The CTI Cisco page provides additional options for Cisco implementations. This page appears only when Cisco is selected as the CTI Vendor on the General Configuration page and the chosen CTI type is Applet.

**Peripheral ID**

Enter the switch (ACD) that the current configuration uses. The peripheral ID is the alias used to identify a switch within the system.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Secondary Host** and **Secondary Port**

Enter a second ICM server. In most cases, the secondary server is used for fail over and load balancing.

**CTI Server Protocol**

Enables you to set the server protocol for a configuration. Currently, only protocol 6 is supported.

**Heartbeat**

Enter an interval in seconds for the system to send a message to the CTI Server to make sure it is running. To disable the feature, enter *−1*. To enable it, enter a value in seconds. The minimum value is 5 seconds. Refer to the Cisco documentation for any recommendations for this option.

**Agent State Mask**

Enter a combination of agent state masks that the CTI agent wants to receive. For instance, you may set the AGENT_AVAILABLE_MASK when you want the application to receive available AGENT_STATE_EVENT messages. Oracle supports the following masks:

- AGENT_LOGIN_MASK
- AGENT_LOGOUT_MASK
- AGENT_NOT_READY_MASK
- AGENT_AVAILABLE_MASK
- AGENT_TALKING_MASK
- AGENT_WORK_NOT_READY_MASK
- AGENT_WORK_READY_MASK
- AGENT_BUSY_OTHER_MASK
- AGENT_HOLD_MASK

**Note.** This value can be overridden by a Cisco system-level mask.

Refer to your Cisco documentation for more information.

**Call Control Mask**

The PeopleSoft system does not currently use this mask in the CTI application. This option is reserved for future use.

Refer to your Cisco documentation for information about this mask.

**Note.** This value can be overridden by a Cisco system-level mask.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Call Progress Mask**

Displays any unsolicited call event messages that you want your application to receive. For instance, you can enter the events that a particular application depends on, such as Begin_Call, CALL_DELIVERED_EVENT, and so on. You can opt to skip events that the application does not depend on, and this can reduce network traffic.

The PeopleSoft system uses the following masks:

- BEGIN_CALL_MASK
- END_CALL_MASK
- CALL_DATA_UPDATE_MASK
- CALL_FAILED_MASK
- CALL_DELIVERED_MASK
- CALL_ESTABLISHED_MASK
- CALL_HELD_MASK
- CALL_RETRIEVED_MASK
- CALL_CLEARED_MASK
- CALL_CONNECTION_CLEARED_MASK
- CALL_ORIGINATED_MASK
- CALL_CONFERENCED_MASK
- CALL_TRANSFERRED_MASK
- CALL_DIVERTED_MASK
- CALL_SERVICE_INITIATED_MASK

**Note.** Modify this value with caution. If you mistakenly elect to skip an event on which an application depends, the application can fail.

This value can be overridden by a Cisco system-level mask.

**Call Variable Mask**

This option is intended for future use. It is not currently implemented for use with PeopleSoft applications.

Refer to your Cisco documentation for information about this mask.

**Note.** This value can be overridden by a Cisco system-level mask.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Service Request Mask** Enables you to adjust CTI service masks. The PeopleSoft system uses the following masks:

- CTI_SERVICE_CLIENT_EVENTS
- CTI_SERVICE_CALL_DATA_UPDATE
- CTI_SERVICE_CLIENT_CONTROL

If you modify this option, do not enter a value lower than *7*.

Refer to your Cisco documentation for more information about this option.

**Note.** This value can be overridden by a Cisco system-level mask.

**Transfer Conference Setup Mask** Enter the valid ways the application can be configured for a transfer or conference call.

Refer to your Cisco documentation for more information about this option.

**Other Feature Mask** Enables you to select the other features supported by an application.

Refer to your Cisco documentation for more information about this option.

Two additional parameters are available to configure the Cisco expanded call context variable. Configure these expanded call context variables in the Cisco middleware:

- user.PS.referenceID: CTI internal use only.

  The console uses this parameter to identify a call.

- user.PS.OutboundContext: This variable, a string type, is attached to an outbound call.

  This variable can be used to associate an outbound call with the desired context.

# Configuring PeopleSoft CTI Queues and CTI Agents

To configure CTI queues and agents, use the Queue Configuration (PT_CTI_QUEUE) and CTI Agent Configuration (PT_CTI_AGENT) components.

This section discusses how to:

- Configure CTI queues.
- Configure CTI agents.
- Use the Phone Book page.
- Personalize he agent console.
- View information about the Agent Info page.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

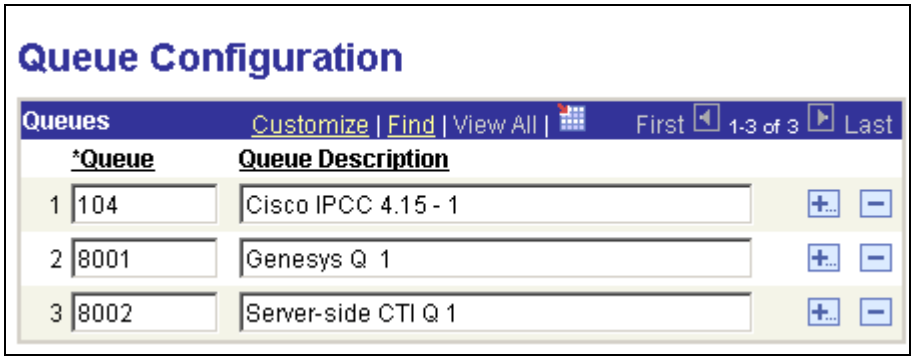
## Pages Used to Configure CTI Queues and CTI Agents

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Queue Configuration | PT_CTI_QUEUE | PeopleTools, MultiChannel Framework, CTI Configuration, Queue | Configure CTI queues. |
| CTI Agent Configuration | PT_CTI_AGENT | PeopleTools, MultiChannel Framework, CTI Configuration, Agent, CTI Agent Configuration | Configure CTI agents. |
| Phone Book | PT_CTI_PNAGENT | PeopleTools, MultiChannel Framework, CTI Configuration, Agent, Phone Book | Use an agent phone book. |
| Personalization | PT_CTI_AGENTGUI | PeopleTools, MultiChannel Framework, CTI Configuration, Agent, Personalization | Personalize agent console and pop-up windows. |
| Agent Information | PT_CTI_AGENTINFO | PeopleTools, MultiChannel Framework, CTI Configuration, Agent Information | View information about the current agent. |

## Configuring CTI Queues

Access the Queue Configuration page using the following navigation path:

PeopleTools, MultiChannel Framework, CTI Configuration, Queue




The Queue Configuration page having the following editable fields: Queue and Queue Description

Use this page to add queues for agents.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Queue** Enter the directory number identifying an ACD group. Calls to a group are distributed to ACD agents belonging to that group, according to ACD algorithms.

**Note.** Queue names can be alphanumeric.

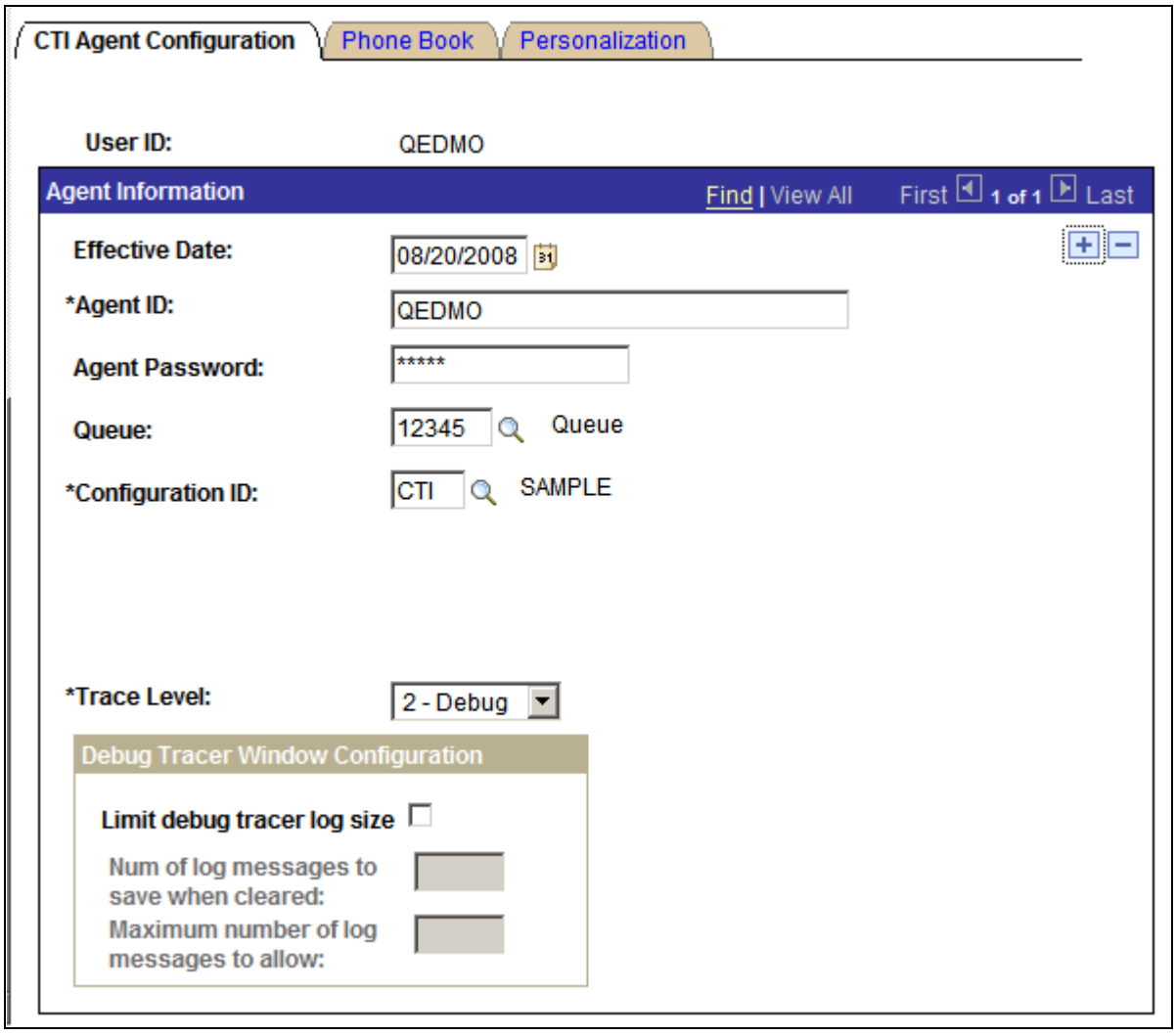
**Queue Description** Enter a brief description of the queue.

See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Defining PeopleSoft MCF Queues for a Third Party, page 217.

## Configuring CTI Agents

Access the CTI Agent Configuration page using the following navigation path:

PeopleTools, MultiChannel Framework, CTI Configuration, Agent, CTI Agent Configuration




CTI Agent Configuration page


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Configuring CTI agents involves the CTI Agent Configuration page, the Phone Book page, and the Personalization page.

**User ID** Displays the PeopleSoft user ID of the agent.

### *Agent Information*

**Effective Date** Enter the date on which the current configuration should become active.

**Agent ID** Enter a user ID for the agent within the switch.

**Agent Password** Enter the password that the agent uses to sign in to the phone, if any.

**Queue** Enter the name of the queue that you want to assign to an agent. Use the Queue Configuration page to associate a queue with a directory number identifying an ACD group.

**Configuration ID** Select the name of the configuration that you want to associate with the agent. The configuration ID is the name of the configuration that you created using the CTI Applet or CTI Non-Applet page.

**Application User Name/Password** These fields are applet-specfic and appear only in the applet-based solution. Enter the CTI user name and password of the agent.

**CTI Client Signature**

---

**Note.** This option appears only for Cisco configurations.

---

This control shows the signature of a particular agent. The signature uniquely identifies an agent if you have implemented call monitoring. Typically, this value appears as an email address, such as john.doe@example.com.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Trace Level**

Options are:

- *0 - None:* Disables tracing.
- *1 - Info* (informational): Traces agent actions, such as dialing out, transfers, and so on.
- *2 - Debug:* Used to troubleshoot crashes and other major errors.

  If you ever need to open a PeopleSoft GSC case about a CTI issue, include a level 2 - Debug trace.

If a value other than *0* is selected, a tracer window appears to display activities and events on the chat or MultiChannel Console for debugging purposes.

If you are using the JSMCAPI (non-applet) console, two tracers (a JSMCAPI tracer and a console tracer) appear if trace level 2, Debug, is selected.

If you are using the Java applet console, trace information is written to the browser's Java Console, which must be enabled. In Microsoft Internet Explorer, enable the Java Console by selecting Tools, Internet Options, Advanced, Java, Java Console. View the Java Console by selecting View, Java Console. To clear the console, press C on your keyboard.

In Sun Java, enable Java Console by selecting Tools, Sun Java.

**Note.** Setting the trace level to *2 - Debug* can degrade performance. Unless you are troubleshooting the system, set the trace level to *None.*

**Limit debug tracer log size**

This check box is enabled when the value entered for Trace Level is not *0.* Select this check box to enable the agent to clear the tracer log based on Number of log messages to save when cleared and Maximum number of log messages to allow.

If the check box is cleared, the tracer log will not get cleared and the Number of log messages to save when cleared and Maximum number of log messages to allowfields will be disabled.

**Number of log messages to save when cleared**

Specify the minimum number of recent tracer log messages that should be maintained in the tracer window.

**Maximum number of log messages to allow**

Specify the maximum number of tracer log messages that will be maintained in the tracer window.

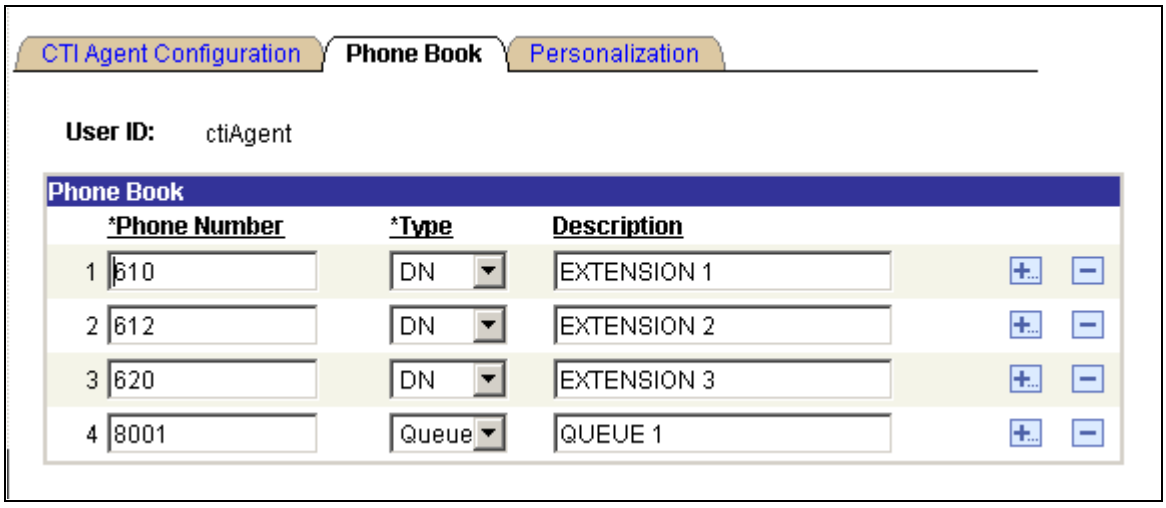
See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Creating PeopleSoft MCF Agents for a Third Party, page 222.

## Using the Phone Book Page

Access the Phone Book page using the following navigation path:

PeopleTools, MultiChannel Framework, CTI Configuration, Agent, Phone Book

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Phone Book page displaying the user ID and having the following editable options: Phone Number, Type, and Description

On this page, you can manage a list of frequently dialed telephone numbers for a *specific* CTI agent. These numbers appear when that agent selects the drop-down list box for a number to dial in the CTI Console. This listing saves the agents from having to manually enter frequently dialed numbers when making outbound calls.

---

**Note.** Phone lists are updated on the CTI Console only after the CTI Console launches. To refresh phone lists, refresh the browser and reactivate the console.

---

**Phone Number** Enter frequently dialed telephone numbers associated with a particular agent.

**Type** Select one of the following options:

- *DN* (directory number): This number identifies a telephone set on a PBX or in the public network. The caller dials this number to establish a connection to the addressed party. The DN can be a local PBX extension (a local DN) or a public network telephone number.
- *Queue* Directory number identifying an ACD queue or group. Calls to a group are distributed to agents belonging to the group, according to ACD algorithms.

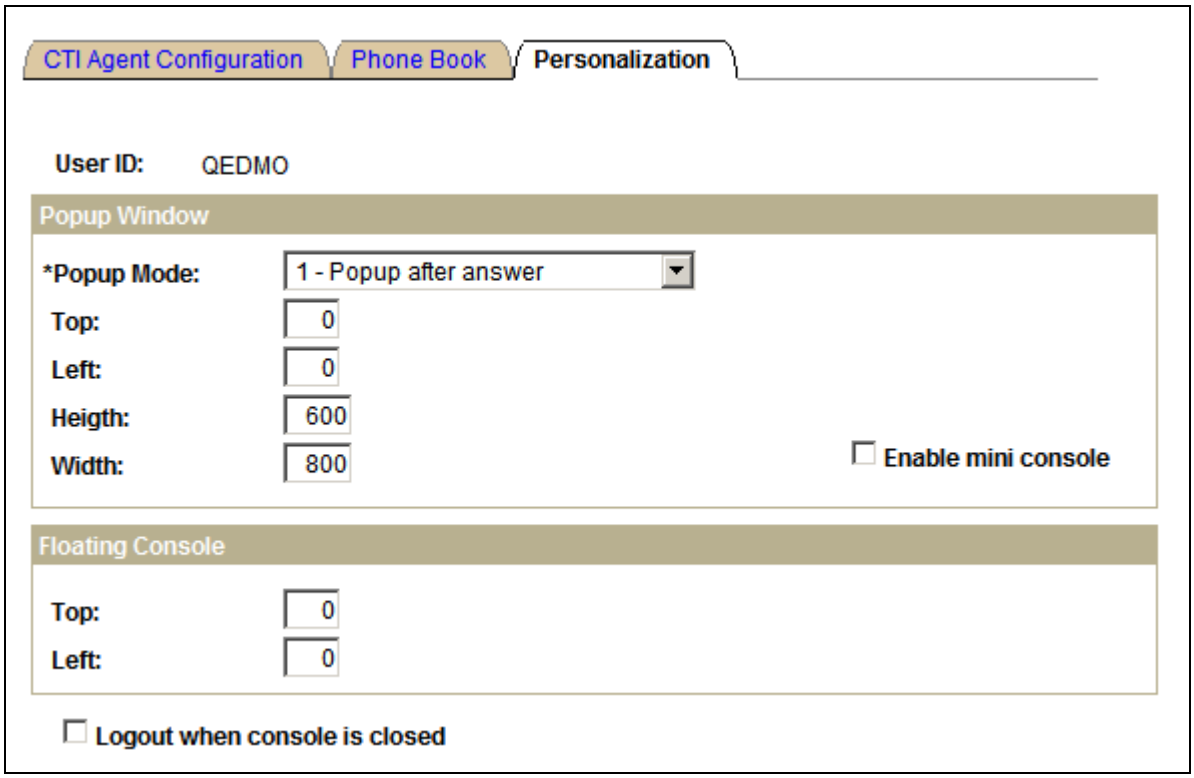
**Description** Add a description for the telephone number.

## Personalizing the Agent Console

Access the Personalization page using the following navigation path:

PeopleTools, MultiChannel Framework, CTI Configuration, Agent, Personalization

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Personalization page

Use this page to personalize timing, size, and position of the pop-up window on the desktop as well as the position of the floating console.

### *Popup Window*

**Popup Mode**

Enables you to configure when the pop-up window appears. You can have it appear after the call is answered, or you can have it appear when a call comes in. If it appears before the call is answered, the agent can determine whether she or he wants to answer the call based on the information that appears in the pop-up window.

The default pop-up mode is *1- Popup after answer*.

In some cases, for example accepting a transfer, although you set the value as *0- Popup when incoming*, you still get the pop-up window after you answer. This is because no call data was attached by the CTI vendor to the incoming event, and consequently the PeopleSoft CTI Console couldn't build the URL for the pop-up window.

For example, assume party A is transferring a call to party B. Party B gets the incoming event, but party B doesn't get a pop-up window. This is because the console did not receive the user data to generate the URL despite the fact that the mode is set to *0*. But after B answers this call, A completes the transfer. Then, B gets the event *partychanged*, receives the user data, and generates the URL and pop-up window.

**Top**

Enter the top position, in pixels. This value is relative to the top of the screen.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Left** Enter the left position, in pixels. This value is relative to the left side of the screen.

**Height** Enter the height of the window, in pixels. The minimum value is 100.

**Width** Enter the width of the window, in pixels. The minimum value is 100.

**Logout when console is closed** This setting depends on CTI vendor support in applet-based consoles, and may not be supported by all vendors. For JSMCAPI (non-applet) consoles and applet consoles with vendor support, select this setting to automatically sign out the agent when the CTI Console is closed.

**Enable mini console** Select to enable the administrator or user to configure the presence of the CTI mini console in the pop-up window. If this check box is not selected, the mini console does not appear in the popup window.

**Note.** The mini console has performance overhead.

### *Floating Console*

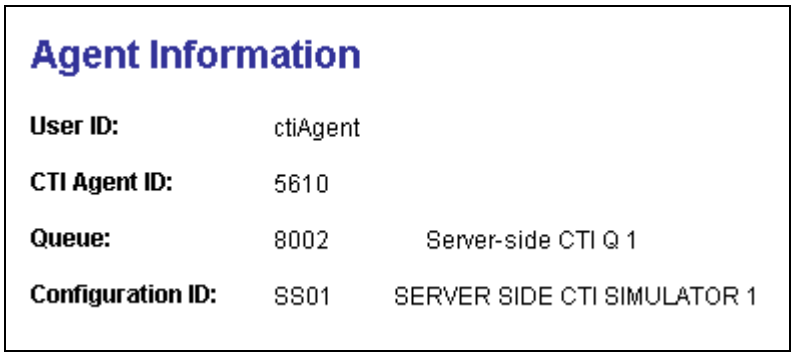
**Top** Enter the top position in pixels. This value is relative to the top of the screen.

**Left** Enter the left position in pixels. This value is relative to the left side of the screen.

## Viewing Information About the Agent Information Page

Access the Agent Information page using the following navigation path:

PeopleTools, MultiChannel Framework, CTI Configuration, Agent Information




The Agent Information page showing the user ID, CTI agent ID, queue, and configuration ID

The Agent Information page is a read-only page that displays the following information about a CTI agent.

**User ID** Displays the agent's PeopleSoft user ID.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| | |
|---|---|
| **Agent ID** | Displays the agent's CTI middleware ID. |
| **Queue** | Displays the queue to which an agent is assigned. |
| **Configuration ID** | Displays the configuration ID associated with the agent. |

# Using Other PeopleSoft CTI Options

You can configure other optional CTI parameters.

This section discusses how to:

- Configure pop-up windows.
- Support single sign-in.
- Log CTI events.
- Implement free seating.

## Configuring Pop-Up Windows

PeopleSoft CTI can launch and populate transaction pages in the following ways:

- Default URL.

  The same URL is used for all calls. The Automatic Number Identification (ANI) is passed in as a parameter to that URL. The ANI identifies the telephone number from which the incoming call originated, and the number may be useful in determining what application data to retrieve. For example, it may be the home phone number of a customer.

- Build the URL from attached Call Data.

  PeopleSoft CTI formats a URL for the browser with a specific target PeopleSoft menu, market, and component. However, this method cannot be used if you are accessing multiple databases through the PeopleSoft Portal.

- iScripts.

  PeopleSoft CTI opens the transaction page using an iScript. This method must be used if you are accessing multiple databases through the PeopleSoft Portal. The iScript communicates with the targeted database to populate the appropriate transaction page with the caller's data.

**Note.** The call ID is passed in the URL string as a variable named *callID.*

**Note.** Browser settings and add-on tools that block pop-up windows can prevent CTI inbound call pop-up windows from appearing. If possible, configure such tools to allow pop-up windows from your site or domain.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Setting Up Genesys for Pop-Up Windows Using Applet Solution*

Unless you choose to use the default URL for your pop-up windows, you must create certain user-defined variables that are attached to each incoming call. These variables are then sent to the PeopleSoft CTI application providing instructions on which PeopleSoft Pure Internet Architecture page appears for the agent.

---

**Note.** The following variables are case-sensitive. Match the case of the variables as shown in this section.

---

**ICType**

Represents the type of PeopleSoft service being called, either a panel (page) or a script.

If the ICTYPE is set to *Panel,* then the following three attached data variables must be set:

- Menu: The name of the PeopleSoft menu containing the destination component.
- Market: The market property of the target component.
- Component: The target component name in the PeopleSoft Application.

If the ICTYPE is set to *Script,* then the following attached data variable must be set:

ICScriptProgramName. This represents the location of the iScript, which is run through the pop-up screen.

**Descr**(description)

Optional data for descriptive purposes only. You can add any descriptive information that might be useful to the agent. This information appears in the CTI Console. For example, you may want the agent to be aware of the customer's priority status, as in Gold Customer.

**Application Specific Data**

Any other attached data keys that are sent to the CTI Console by the Genesys telephony server, such as customer number, are passed to the target application page as parameters.

The parameters are separated from the PeopleSoft URL by a question mark (?). Parameters are separated from other parameters by an ampersand (&). If you are using iScripts, you can use the GetParameter methods to read the parameters in your PeopleCode. If you choose to use ICPanels instead, the parameters are used as the key list.

Refer to your PeopleSoft application documentation for specific instructions on any other attached data keys that must be passed to the CTI Console by Genesys.

### *Setting Up Cisco for Pop-Up Windows Using Applet Solution*

Unless you choose to use the default URL for your pop-ups windows, you must create certain expanded call context (ECC) variables. ECC variables are variables that you define and enable in the Cisco ICM Configuration Manager to store values associated with the call. These variables are sent to the PeopleSoft CTI Console when ICM notifies it of an incoming call, and the console uses the variables to determine which PeopleSoft page to launch for the agent.

Every call data key used in your implementation must be registered in the Cisco ICM, before the PeopleSoft system can receive the attached call data from the IVR.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

The PeopleSoft system recognizes only data keys with *user.PS.* in front of all the PeopleSoft key names. For example,

```
user.PS.Descr
```

The following table contains the call data variables that must be created and registered in the ICM.

**Note.** The following variables are case-sensitive. Match the case of the variables as shown in this section.

| | |
|---|---|
| **ICType** | *user.PS.ICType.* 6 characters. Valid values are *Panel* (page) or *Script.* It represents the type of PeopleSoft service being called, either a panel (page) or a script.<br>If ICType is set to *Panel,* the following variables apply:<br>• user.PS.Component. 20 characters. This variable is required to specify the name of the destination component.<br>• user.PS.Menu. 10 characters. This variable is required to specify the name of the menu within PeopleSoft containing the destination component.<br>• user.PS.Market. 3 characters. This variable is required to specify the name of the market of the destination component.<br>If the ICType is set to *Script,* then the following variable applies:<br>user.PS.ICScriptProgramName. 70 characters. If the ICTYPE is set to Script, then this variable is required to specify the location of the iScript, which is run through the pop-up screen. |
| **Descr** | user.PS.Descr. 30 characters.<br>You can add any descriptive information that might be useful to the agent. This information appears in the control bar. For example, you may want the agent to be aware of the customer's priority status, as in Gold Customer. |
| **Application Specific Data** | Any other variables that the application requires must be prefixed with *user.PS.* PeopleSoft truncates *user.PS.* before passing the parameter to the application. For example,<br>user.PS.App1 (3 characterss)<br>user.PS.App2 (1 characterss) |

**Note.** Typical character sizes for these variables appear in the previous table. The actual size of these variables depends on the components specified by the PeopleSoft application you are using. Refer to your PeopleSoft application documentation for correct size information.

### *See Also*

*PeopleTools 8.51 PeopleBook: PeopleCode API Reference,* "Internet Script Classes (iScript)"


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Supporting Single Sign-In

In the applet-based solution, PeopleSoft CTI offers single sign-in. The console connects to the CTI middleware using the CTI user ID and password retrieved from the PeopleSoft database.

### *See Also*

*PeopleTools 8.51 PeopleBook: Security Administration*, "Implementing Single Signon"

## Logging CTI Events

CTI events can be logged in the CTI Event Log. You can view the logged CTI events by navigating to PeopleTools, MultiChannel Framework, Universal Queue, Administration, CTI Event Log.

To implement CTI event logging, navigate to PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Cluster Tuning, and set the key *log_cti* to *yes.* Use the Notify Cluster page to notify the affected queue to refresh logging parameters.

The CTI event log logs all events and requests related to a session, user, or call connection. It also logs all events related to call and group information.

---

**Note.** Applet-based CTI events are not logged.

---

### *See Also*

Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Tuning Cluster Parameters, page 108

## Implementing Free Seating

When users sign in, they do not need to reenter telephone extensions and other user information if they have used the workstation before and the relevant information for the telephone associated with that workstation has not changed.

The PeopleSoft system enables free seating by maintaining a cookie on the workstation.

# Using the PeopleSoft CTI Sample Pages

To demonstrate an outbound call, use the Sample Pages component (PT_CTI_DEMOOUTB). The CTI sample pages are intended for demonstration purposes only and should not be used in production.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

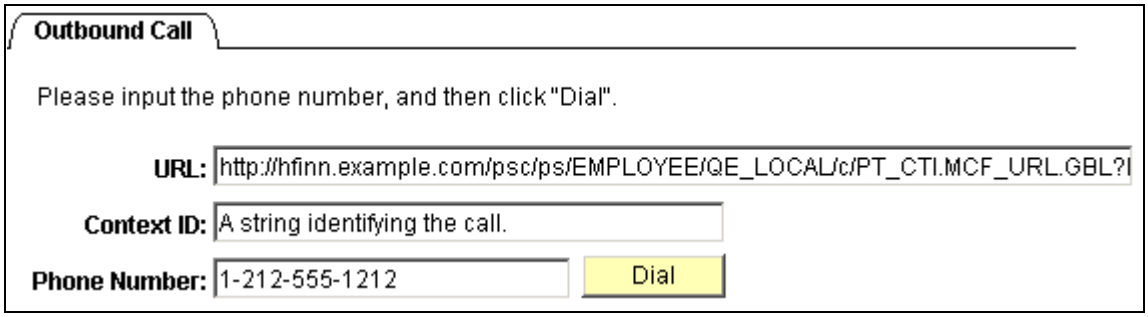
## Page Used to Demonstrate Outbound Calls

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Outbound Call | PT_CTI_DEMOOUTBOUN | PeopleTools, MultiChannel Framework, CTI Configuration, Sample Pages, Outbound Call | Demonstrate making an outbound call. |

## Using the Outbound Call Page

Access the Outbound Call page using the following navigation path:

PeopleTools, MultiChannel Framework, CTI Configuration, Sample Pages, Outbound Call




The Outbound Call page having the following editable fields: URL, Context ID, and Phone Number to dial

The Outbound Call page is an example of how you can customize an application page to enable a user to direct the CTI Console to dial a telephone number displayed on that page. The outbound calling demonstration works only when the CTI Console is enabled and the user has registered with the CTI vendor.

**URL** Enter the URL of a page to display when dialing out.

**Context ID** Enter a string to attach to the call as outbound context.

**Phone Number** Enter the telephone number that you want to dial. This field accepts numeric digits only. Do not enter special characters, such as - (hyphen), . (period), or other separators.

**Dial** Click to dial the telephone number that you entered.

---

**Note.** The CTI Outbound Call sample page may fail when used with the CTI applet console and a Netscape browser if you have specified an authentication token domain for PeopleSoft Pure Internet Architecture.

---


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *See Also*

Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the CTI Sample Console, page 184

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 4

# Using PeopleSoft CTI

This chapter provides an overview of PeopleSoft CTI and discusses how to use PeopleSoft CTI using a CTI server or an integrated third-party routing system.

## Understanding PeopleSoft CTI

This section discusses:

- PeopleSoft CTI.
- Cisco switch considerations for the applet-based solution.

### PeopleSoft CTI

PeopleSoft CTI is a browser-based call-management system that helps call agents work more efficiently with customers. PeopleSoft CTI integrates Oracle-validated third-party CTI systems and your PeopleSoft applications. It exchanges data between the CTI system and your PeopleSoft applications so that the system automatically fills PeopleSoft transaction pages with the appropriate customer information—the information related to the caller.

With PeopleSoft CTI, you can perform the following tasks:

- Operate two lines or two extensions.
- Answer incoming calls.
- Release calls.
- Put a caller on hold.
- Monitor call status.
- Access PeopleSoft applications.
- Transfer callers.
- Initiate conference calls.
- Place an outbound call.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Note.** PeopleSoft CTI supports Oracle-validated third-party CTI systems. In the following sections, when the phrase *your CTI middleware* or *CTI vendor* appears, assume that it refers to Oracle-validated third-party CTI systems, as appropriate for your installation.

For a list of partners that offer CTI middleware integrations, refer to Oracle Validated Application Integrations – Find a Partner Solution, http://www.oracle.com/partnerships/isv/integration/search.html

Because a CTI console developed with the JavaScript MultiChannel Application Programming Interface (JSMCAPI) can be customized, features described in this chapter may not appear or may act differently.

#### *The CTI Interface*

The PeopleSoft CTI interface is incorporated *within* the MultiChannel Console.

#### *See Also*

Chapter 3, "Configuring PeopleSoft Computer Telephony Integration," page 15

Chapter 10, "Managing Tasks and Using Chat in PeopleSoft MultiChannel Framework," page 151

## Cisco Switch Considerations for the Applet-Based Solution

The following list presents the limitations of the Cisco Systems Internet Protocol (IP) Contact Center switch:

- MAKE_CALL is supported only when the agent is in the Not Ready state.
- Consultative and blind transfers are supported.

  Placing a call on hold and making a new call and completing the transfer is not supported due to limitations in the Call Manager.
- Hold and retrieve are supported, but not during a consultative call because it disrupts the consult relationship in Call Manager.
- Only the conference initiator can add parties to the conference.
- Cisco does not support two applications managing the same call.

  Therefore, you cannot run a Cisco softphone and the PeopleSoft CTI console on the same workstation.

**Note.** Refer to your CTI vendor documentation for further information about any limitations imposed by a particular switch. In some cases, the switch limitations may actually be due to the CTI server, so be sure to refer to any server limitations that may also impose limitations on the switch.

# Using PeopleSoft CTI

This section discusses how to:

- Get started.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Use the CTI console.
- Select call actions.
- Answer a call.
- Transfer a caller.
- Initiate conference calls.
- Work with the hold status.
- Disconnect a caller.
- Switch agent ready status.
- Dial an outbound call.
- Complete a call.
- Use hot keys.

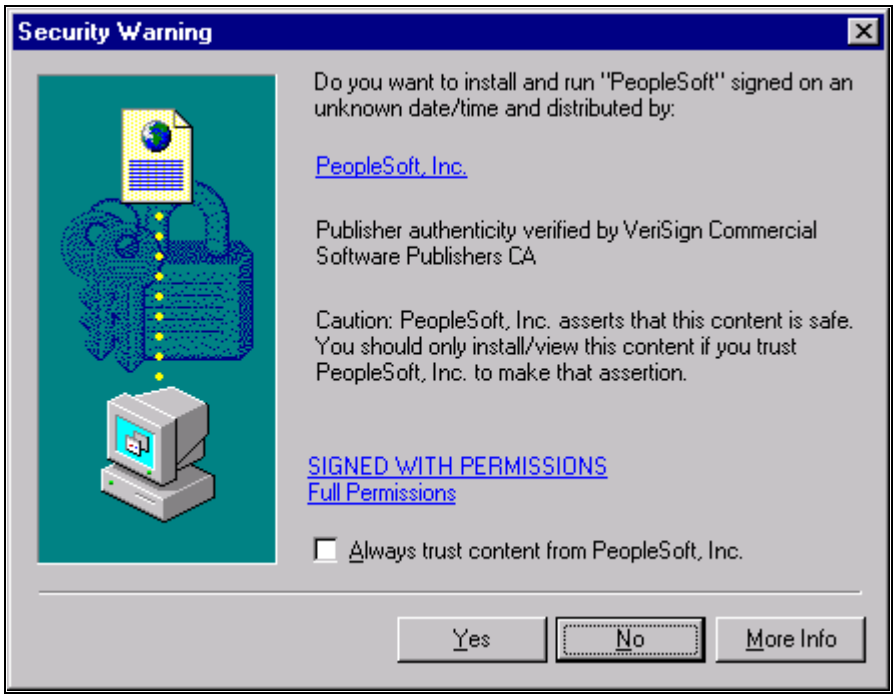
## Getting Started

Getting started with PeopleSoft CTI is a three-step process that involves:

- Signing in to the PeopleSoft system.
- Specifying your extension information.
- Connecting to your CTI middleware.

The first time that you access PeopleSoft CTI using the applet-based solution, a Security Warning dialog box *may* appear prompting you to trust information from PeopleSoft. Click the Yes button.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Security Warning dialog box asking you whether you want to trust information from PeopleSoft

The Security Warning dialog box may not appear if:

- You have previously accepted PeopleSoft applets as trusted.
- The system is not correctly installed.

---

**Note.** If you have any questions or concerns about this warning, contact your system administrator.

---

To sign in to PeopleSoft CTI:

1. On the PeopleSoft sign-in page, enter your PeopleSoft user ID and password as you normally do to sign in to the PeopleSoft system.
2. Click the MultiChannel Console link in the universal navigation header.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

3. On the CTI control bar, set your configuration by performing the following actions:

   - Ensure that your agent ID appears beneath the CTI Agent ID label.
   - Ensure that the appropriate queue name appears beneath the Queue label.

     Queues are discussed in a subsequent section.
   - If you are signing in from a workstation with a different extension, enter the current extensions in the Extension field.
   - The Extension Type field indicates whether an agent can receive calls by way of a queue or just from a directory number (DN).

     If this value is set to *Queue,* the console offers options that are queue-specific, such as being able to log on to a queue.
   - Click the Activate button to sign in to your CTI middleware.

     The CTI console appears.

## Using the CTI Console

Change any registration parameters if necessary and click Set or Activate (depending on your version). After you do so, the CTI interface appears, which looks similar to the following:

---

**Note.** To launch server-side CTI and PeopleSoft MultiChannel Framework (MCF) console using Netscape, go to Tools, Popup manager, Allow popups and add your URL.

---

**Note.** Note that the PeopleSoft CTI interface operates *within* the MultiChannel Framework Console. CTI users will see the MultiChannel Console only if their user ID belongs to a role that has MCF_AGENT or MCF_SUPR real-time event notification (REN) permissions, and WEBLIB_MCF web libraries in their role's permission list.

---

The CTI Console

---

**Note.** Do not open two consoles for the same user on a single machine. Doing so can result in an error.

---

**(register)**

Select to register and unregister with the CTI middleware. The button acts as a toggle switch. When the green check mark appears, you are registered; when the red *X* appears, you are not registered.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Lines** Select the option to the left of the telephone icon to activate the associated line. The icon that you select determines the active line. All call actions that you choose apply only to the active line. The color of the icon reveals the activity on the line. The colors are:

- Blue: Inactive.
- Red: Ringing.
- Green: On call.
- Yellow: On hold.

**Select Call Action** Select actions related to calls, including *Answer*,*Hold*, and *Transfer*. Which call action options appear depends on your current status. For instance, the Hold call action is valid only while you are on the line with a caller. Call action options are discussed in detail in the following section.

**Status** Displays the agent status, as in *Ready* or *Not Ready* (to receive calls). It can also show *Ready/DND* (do not disturb) if the agent is not using a queue.

**Q** (queue) If the agent belongs to an automatic call distributor (ACD) group, the group appears here.

**Messages** Informational messages appear in the right corner of the CTI Console. Examples of such messages are *Dialing*,*Connected*, and *Released*. These messages do not persist.

**Note.** Error messages appear in separate windows.

**Call Duration** The system tracks the amount of time spent on calls for each line.

**Incoming Call Information** Displays the string associated with the Descr call variable. For example, *Gold Customer*.

**Note.** In the applet-based solution, some CTI systems will automatically sign out the CTI agent when the CTI console is closed. For CTI systems in which this is not automatic, the option can be set on the agent Personalization page (select PeopleTools, MultiChannel Framework, CTI Configuration, Agent, Personalization.)

## Selecting Call Actions

The Select Call Action drop-down list box contains all of the options that you have for handling calls. Depending on the status of the agent or the telephone line, certain selections from the drop-down list box are not available.

After you select a call action, two buttons, Go and Cancel, appear to the right of the Select Call Action drop-down list box. Go performs the selected call action, and Cancel stops the call action that you carried out.

The following list describes the call actions:


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Dial** When you are in Agent Ready or Agent Not Ready mode, you can call another party.

**Answer Call** This button appears and flashes when an incoming call is waiting to be answered.

**Transfer Mute** Transfers the caller to the desired number without speaking to the intended recipient. A dialog box appears enabling you to enter the extension of the person to whom you want to transfer the caller.

**Note.** In the applet-based solution, this action is not supported on Aspect switches.

**Transfer** Transfers the caller to the desired number. You have the opportunity to speak with the recipient before transferring the caller. A dialog box appears enabling you to enter the extension of the person to whom you want to transfer the caller.

**Conference** Enables you to add one or more individuals to your call.

**Hold** Places the caller on hold.

**Retrieve** Takes the caller out of *Hold* status. This option appears only when the caller is on hold.

**Release** Disconnects the caller. This option is available as a selection only after a call is answered.

**Change** When an agent status is unknown, the system sets the status to *Change* so that the user can set the status manually to match the status for the telephone.

**Note.** You can transfer to or initiate a conference call with individuals who are not enabled to access PeopleSoft CTI. Their phone rings, but remember that the pop-up window showing customer data does not appear.

In the applet-based solution, the following call actions are available depending on whether you are using a DN or Queue configuration:

**Queue** Select from:

*Log on:* Enables the agent to log an extension on to a queue.

*Ready:* Indicates that the agent is ready to receive incoming calls. This option is available when the extension is associated with a queue and the status bar reads Agent Not Ready.

*Not Ready:* Stops incoming calls. This option is available when the extension is associated with a queue and the status bar reads Agent Ready.

*Log off:* Enables the agent to log an extension off a queue. When logged off, the agent is no longer participating in the queue.

**DN** This option is available when the extension is not associated with a queue.

*Ready:* Indicates that the agent is ready to receive incoming calls.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

In the adapter solution, the availability and meaning of the call actions are determined by the third-party CTI vendor. Consult your CTI vendor for details.

## Answering a Call

After you have signed in to PeopleSoft CTI, you can receive calls. For each incoming ACD call to your extension, the telephone extension icon turns red. After you have accepted a call, the system does not send you more incoming calls until you have completed the current call.

To answer a call:

1. Select the radio button to the left of the telephone icon that has turned red.

   The *Answer* option is automatically selected as the current option in the drop-down list box when an incoming call arrives.

2. Click Go.

   The pop-up browser launches with the appropriate PeopleSoft transaction page displayed. The system determines which page to display based on caller information sent by the CTI middleware.

   After you have answered a call, you enter the not available status.

If an agent erroneously clicks Cancel instead of Go when accepting an incoming call, the agent can recover the call by:

1. Selecting a call option.

2. Selecting the line that is receiving the incoming call.

---

**Note.** If CTI screen pop-ups are configured, the CTI console attaches a CTI miniconsole to the screen pop-up page. This miniconsole takes a few seconds to initialize before it can be used.

In the Javascript console, you can configure whether the miniconsole appears.

---

## Transferring a Caller

Occasionally, you need to transfer callers to other agents. PeopleSoft CTI supports two types of transfers:

- Transfer Mute: This option enables you to transfer a call without speaking to the target agent before transferring the call.

- Transfer: This option is also known as a consultative transfer, which means that you consult with the target agent before transferring the call.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Note.** You can always transfer or invite users to a conference call even if the called party is not CTI-enabled. The non-CTI-enabled users do not get pop-up windows, but their phones still ring.

When you initiate a transfer or conference call on the Cisco system using the applet-based solution and the contacted party does not answer, wait until the system notifies you of the unsuccessful connection before attempting another action. This wait can be 20 to 30 seconds.

In the applet-based solution, Cisco ICM does not notify client applications about some call events during two-step transfers or conferences. As a result, the state of the PeopleSoft CTI Console may not stay synchronized with that of the teleset, especially if the teleset is used to initiate or complete these call actions. The CTI console should automatically resynchronize with the teleset when the call is completed.

To perform a transfer mute:

1. Select the appropriate telephone line.

   The selected line must be green.

2. From the Select Call Action drop-down list box, select *TransferMute*.

3. In the Phone No. edit box, select the number that you want to dial.

   The drop-down list box contains all of the numbers from the shared phone book and agent phone book. If the number that you want to dial does not appear, click Dial other number and manually enter the number.

4. Click Go.

   This connects the caller to the new agent and releases your line.

   The system prompts the recipient of the transfer that it is transferring a call from your extension. When the recipient accepts the transfer, the PeopleSoft page connected to the caller's case opens as it did when you first received the call.

To perform a transfer (consultative):

1. Select the appropriate telephone line.

   The selected line must be green.

2. From the Select Call Action drop-down list box, select *Transfer*.

3. In the Phone No. edit box, select the number that you want to dial.

   The drop-down list box contains all of the numbers from the shared phone book and agent phone book. If the number that you want to dial does not appear, click Dial other number and manually enter the number.

4. Click Go.

   When you use one extension with two lines, the outbound call you make to the agent to whom you are transferring the incoming call gets initiated on your second line and the incoming call gets placed on hold. When the outbound call is established, you can consult with the recipient and place that call on hold. To complete the transfer, you need to return to the first line and select *Complete* and click Go. This action releases the call on the second line and transfers the call on the first line to the recipient of the transfer.

   When you use two extensions, each with one line, the CTI Console does not have access to the outbound call to the intended recipient of the transfer. When the outbound call is established, you can consult with the recipient. You do not have to toggle between the two lines, and you cannot put the recipient on hold.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

5. To complete the transfer, select *Complete* and click Go.

## Initiating Conference Calls

If you need the assistance of other agents to answer a caller's questions, you can use the conference feature to include the appropriate agents on a call.

To initiate a conference call:

1. Select the appropriate telephone line.

   The selected line must be green.

2. From the Select Call Action drop-down list box, select *Conference.*

3. From the drop-down list box of all numbers from the shared and agent phone books, select a number to be dialed.

   If the number is not there, select *Dial other number* to access an edit box and enter the number to be dialed.

4. Click Go.

   The system notifies the target agent of the incoming call (conference). The PeopleSoft page associated with the caller's case opens for the target agent as it did for you when you first received the call.

   This feature depends upon two parameters set up by the administrator:

   - Default Screen Pop-up URL.
   - Personalization: Screen Pop-up Mode.

     If the default URL for screen pop-up is set and the screen pop-up mode is 0 (pop up when incoming) for the second agent, the agent gets the screen pop-up as soon as the call is transferred.

     If the screen pop-up is set to 1 (pop up after answer), the screen pops up only after the first agent completes the transfer or conference call. However, if the default URL for the screen pop-up is not set, then whether the mode is 0 or 1 doesn't matter. In that case, the second agent gets the screen pop-up only after the first agent completes the transfer or conference call.

     When you use one extension with two lines, the outbound call that you make to the agent to whom you are inviting to the conference gets initiated on your second line and the incoming call gets placed on hold. After the outbound call is established, you can consult with the third party, and place that call on hold. To complete the conference, you need to return to the first line and select *Complete* and click Go. This action releases the call on the second line and starts the conference on the first line.

     When you use two extensions, each with one line, the CTI Console cannot access the outbound call to the third party. When the outbound call is established, you consult with the target agent. You do not have to toggle between the two lines, and you do cannot put the recipient on hold. To start the conference, select *Complete* and click Go.

5. After consulting with the target agent, select *Complete* from the Select Call Action drop-down list box, and click Go.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Working with the Hold Status

Putting calls on hold and retrieving calls on hold is likely to be the call action that you perform most.

To place a call on hold:

1. Select the appropriate telephone line.

   The selected line must be green.

2. From the Select Call Action drop-down list box, select *Hold.*

3. Click Go.

To retrieve a call on hold:

1. Select the appropriate telephone line.

   The selected line must be green.

   The retrieve option is automatically selected as the current option in the drop-down list box when a call is on hold.

2. Click Go.

## Disconnecting a Caller

After you have finished a call, you need to release the call.

To release a call:

1. Select the appropriate telephone line.

   The selected line must be green.

2. On the console control bar, select *Release* from the Lines drop-down list box.

3. Click Go.

   If you are using the applet-based solution, the system automatically places you in wrap-up mode, which enables you to complete any remaining work before accepting more incoming calls. Technically, when you are in wrap-up mode, your status is *Agent Not Ready.*

   If you are using the adapter-based solution and you have configured wrap-up mode in your CTI middleware, the console can automatically place you in a *Not Ready* or *Work Not Ready* state, which enables you to complete any remaining work before accepting more incoming calls.

   When you are ready to accept incoming calls, select *Agent Ready.*

## Switching Agent Ready Status

Your agent status determines whether you can receive incoming calls.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

In the applet-based solution, agent statuses include:

- *Agent Ready*

  To activate *Agent Ready* status:

  From the Select Call Action drop-down list box, select *Agent Ready.*

  When you are ready, the system routes incoming calls to your extensions.

- Do Not Disturb

  To activate Do Not Disturb status:

  From the Select Call Action drop-down list box, select *DND.*

  With Do Not Disturb, your extensions do not accept incoming calls.

  **Note.** This status is not available to agents associated with an ACD queue.

- *Agent Not Ready*

  To activate *Agent Not Ready* status:

  From the Select Call Action drop-down list box, select *Agent Not Ready.*

  This status is typically used when agents are at their desks, but temporarily unable to receive calls. While you are not ready, the system routes calls to other available agents.

  **Note.** This status applies only to agents associated with an ACD queue.

**Note.** In the adapter-based solution, the adapter provider is responsible for determining what states and statuses are available in the drop-down list boxes, as well as the meaning of those states.

## Dialing an Outbound Call

You can use PeopleSoft CTI to place a call while the agent status is either *Agent Ready* or *Agent Not Ready.*

**Note.** If you have multiple extensions assigned to you, *do not* call one of your extensions from the other.

To place an outbound call:

1. Select your status from *Agent Ready* or *Agent Not Ready.*
2. Select the option next to the telephone icon representing a free line.

   For example, if you had a customer on hold on one line, you would select the icon for the second line.
3. From the Select Call Action drop-down list box, select *Dial.*
4. From the drop-down list box showing all numbers from the shared and agent phone books, select a number to be dialed.

   If the number is not there, select Dial other number to get an edit box and enter the number to be dialed.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

5. Click Go.

   As with any other call you receive, you can access all the call actions for calls that you initiate. You can transfer the person that you've called, place the line on hold, or initiate a conference with another party.

   ---

   **Note.** If you are using the applet-based solution, the system places you in *Agent Not Ready* status until you release the call. If you are using the adapter-based solution, you must configure your CTI middleware call actions.

   ---

   When you make an outbound call:

   - You can specify a URL to display a page.

     When you display the page, the outbound call data is also attached to the URL so that an application can collect information such as the automatic number identification (ANI), dialed number identification service (DNIS), and so on.

   - A context ID (such as a customer case number or an invoice number) is attached to the call data.

     This allows PeopleSoft application context to be passed on to CTI middleware, where it can be stored. This context can be used to establish a relationship between the outbound call and the application context when the call was made.

#### *See Also*

Chapter 3, "Configuring PeopleSoft Computer Telephony Integration," Creating a List of Frequently Dialed Phone Numbers, page 25

## Completing a Call

When you disconnect or release a call from the miniconsole, the system disconnects you from the CTI middleware, and the miniconsole becomes disabled (unavailable for entry). However, the PeopleSoft page remains active so that you can finish updating information if needed.

The availability of an agent to accept incoming calls depends on how the system administrators of the agent have set up the agent's CTI middleware.

## Using Hot Keys

Hot keys are combinations of keyboard buttons that you can press instead of using a mouse. To help you easily select options, PeopleSoft CTI offers the following hot keys:

| *Hot Key* | *Description* |
|---|---|
| Alt + R | Registers your phone extensions with the CTI middleware. |
| Alt + 1 | Makes Extension 1 the active line. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

| *Hot Key* | *Description* |
|---|---|
| Alt + 2 | Makes Extension 2 the active line. |
| Alt + S | Presents a list of applicable call actions for you to select. |
| Alt + P | Presents a list of frequently called telephone numbers for you to select. |
| Alt + G | Enables you to perform a call action. |
| Alt + C | Enables you to cancel a call action. |
| Alt + Z | Enables you to check agent status. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 5

# Configuring REN Servers

This chapter provides overviews of real-time event notification (REN) servers and Secure Sockets Layer (SSL) enabled REN servers and discusses how to:

- Configure REN server security.
- Configure REN servers.
- Configure REN server and SSL-enabled REN server clusters.
- Configure a reverse proxy server with a REN server.

## Understanding REN Servers

This section discusses:

- REN server failover, scalability, and security configuration.
- REN server failover.
- REN server clusters.

The REN server, an application server domain process, is essential to PeopleSoft MultiChannel Framework (MCF) architecture. MCF events are sent to REN servers, which deliver them to recipients of those topics.

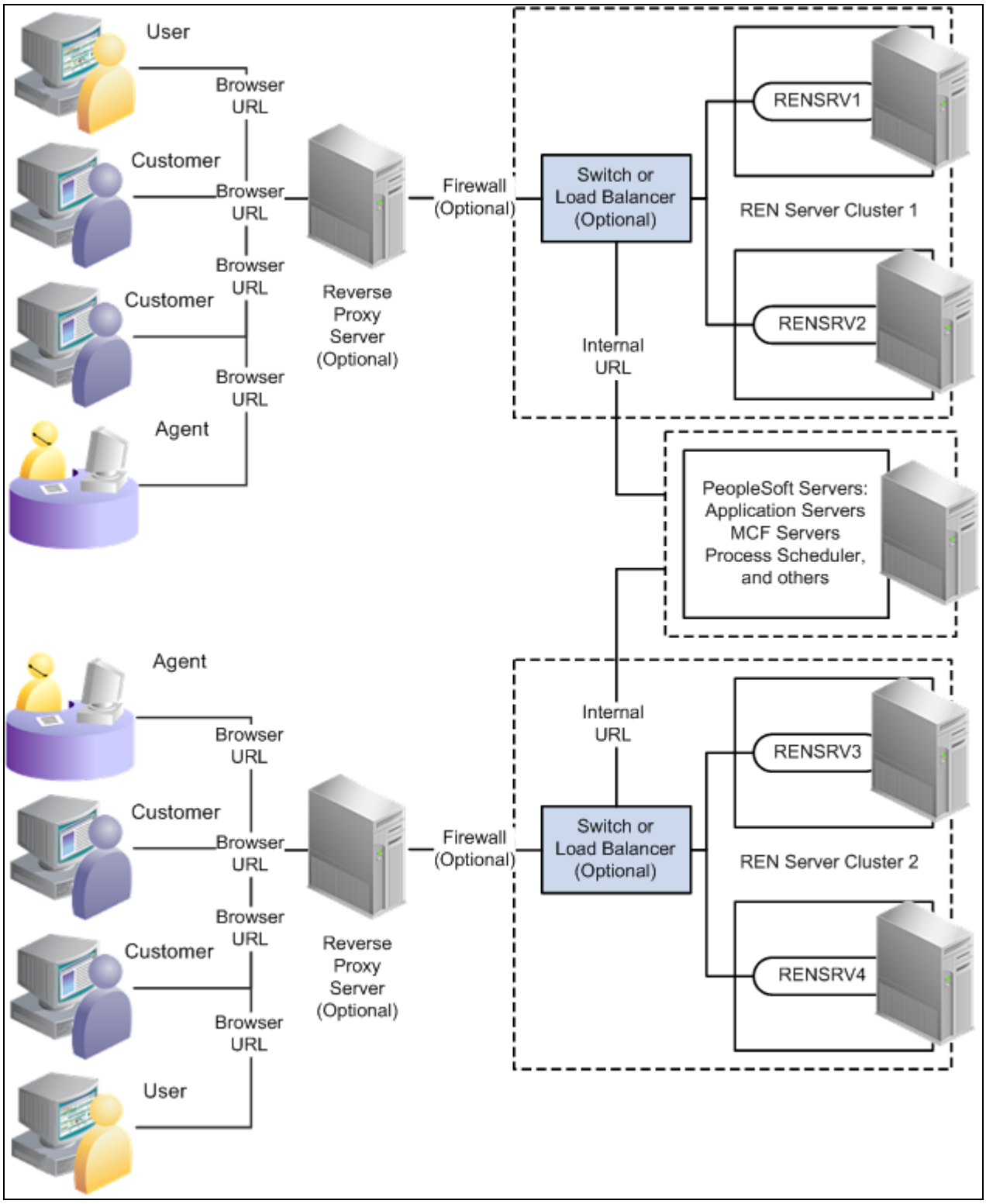
REN servers are also used by other PeopleSoft applications to push event notifications to users, such as the Reporting Window output option and the Optimization Progress Window.

The REN server is a modified web server using the HTTP 1.0 or 1.1 communications protocol. Communication with MCF server processes and browser windows is bidirectional because they maintain persistent connections to the REN server. Events can be sent proactively to browser windows without polling or page refreshes.

### REN Server Failover, Scalability, and Security Configuration

REN servers can be configured to support both failover and scalability, and should be protected with firewalls and appropriate security measures, as illustrated in the following diagram:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




REN server configuration example


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## REN Server Failover

Although the REN server is integrated into an application server domain, it is not a standard PeopleTools server process (it has no database connection) and therefore has a separate failover mechanism. Two scenarios exist for failure recovery:

- For a standalone REN server, Oracle Tuxedo restarts the server if it fails.

  MCF servers and consoles reconnect to the REN server. However, any active browser sessions (such as MCF chat) are interrupted until a connection can be reestablished between the chat console and the restarted REN server.

- For clustered REN servers, each REN server in the cluster is a peer that mirrors the current state.

  This configuration has two advantages over a standalone REN server:

  - Clustered REN servers guard against hardware failure (provided that the clustered REN servers are on different host machines).

  - Active browser sessions are not lost.

## REN Server Clusters

You can configure a REN server cluster with only one REN server member. However, a REN server cluster that is configured with two or more REN servers provides failover.

All REN servers in a cluster mirror each other and appear to external processes as a single URL. The REN server cluster must have an HTTP load balancer or switch as its front end. All connections with browsers and application server processes address the front end's URL. The load balancer should use an active standby content-switching rule to route all traffic to a designated REN server in the cluster. The front end selects an alternate member of the cluster only when the designated REN server fails to respond.

The REN server cluster maintains mirrored state in all members by relaying events with HTTP messages. The REN server cluster therefore does not address scalability issues. Clustering REN servers does not improve performance and may increase processing overhead and internal network traffic. The internal HTTP connections between cluster members should be high speed for best performance. Because of the overhead involved in synchronous cluster members, each member of a cluster can handle less load than a REN server in a cluster with only one REN server.

---

**Note.** In an environment in which multiple REN servers exist within a single cluster, the primary REN server sends synchronization data to the other members of the cluster. If any of these synchronization messages fail, then the primary REN server retries up to cluster_retry_count times. The minimum value of this parameter, cluster_retry_count, in psrenconfig.txt is 0, which means that the REN server does not retry.

---

If a REN server crashes, it does not rejoin the cluster because it would not be synchronized with the other clustered REN servers. The entire cluster must be shut down and rebooted to restore all members back to full participation.

Incoming cluster requests must eventually route to the front end's HTTP address. Queue servers and application servers use the cluster URL, which is typically set to be the URL of the front end. Browser clients make requests using the browser URL, which may be set to the front end, or to a server that proxies to the load balancer. If browser transactions are encrypted with SSL, then the browser URL is an HTTPS address to a reverse proxy server or SSL accelerator.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Note.** If you use SSL between the browser and REN server, then you must use a reverse proxy server or SSL accelerator, unless you have configured an SSL-enabled REN server.

# Understanding SSL-Enabled REN Servers

You can enable a secure channel of communication between the clients and the REN server by enabling SSL on the REN server using openssl. The SSL protocol runs above Transmission Control Protocol/Internet Protocol (TCP/IP) and below higher-level protocols, such as HTTP and IMAP4. By using TCP/IP on behalf of higher-level protocols, openssl allows an SSL-enabled server to authenticate itself to an SSL-enabled client, a client to authenticate itself to a server, and both machines to establish an encrypted connection.

This section discusses:

- Installing digital certificates.
- Authenticating sever and client.
- Performance and scalability for SSL-enabled REN servers.

## Installing Digital Certificates

REN servers require digital certificates to work in SSL mode. The servers pick up the certificates from the PeopleTools database. The certificates must be imported into PeopleTools database from PeopleTools, Security, SecurityObjects, Digital Certificates. Certificates that are installed in the database will have a unique combination of certificate type and alias.

The certificate type that is used for the server should be of the type CERT, and the alias is <machine name>.<domain name>. When the certificate is configured with a unique alias name, it should be associated with the REN server that is SSL-enabled. The REN server loads its server certificate from the database at the start-up.

See Appendix B, "Installing Digital Certificates for REN SSL," page 589.

## Authenticating Server and Client

For server authentication, the server sends its certificate to the client as a part of the SSL handshake and the client authenticates by verifying the Certificate Authority (CA) of the certificate against its trusted keystore. When the REN server is configured for SSL, all clients must trust the CA of the server certificate to participate in a successful communication.

Client authentication verifies the clients's authenticity to participate in a communication with the server. When the REN server is configured for client authentication, all clients must supply a valid client certificate to participate in a successful communication.

All clients must use the REN cluster's HTTPS URL to communicate in the SSL mode. If the REN server is SSL only, access is denied to any client trying to communicate with a HTTP URL port. The browser-based clients, the application server client, and the REN Java clients should be configured appropriately to communicate with an SSL-enabled REN server.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Performance and Scalability for SSL-Enabled REN Servers

During an SSL transaction, the handshake is an added overhead that occurs. However, for every transaction, the handshake is done once to authenticate the server and the client. After authentication, the data is digitally signed, encrypted, and exchanged on an established session. For each console, authentication establishes a session only once, and no subsequent transactions inherit any overhead of authentication.

# Configuring REN Server Security

This section provides an overview of REN server security configuration and discusses how to define permission lists for REN server access.

## Understanding REN Server Security Configuration

Protect the REN server behind firewalls. A reverse proxy server can be used between browser clients and the REN server. Browser sessions can be SSL-encrypted by means of a reverse proxy server or hardware SSL accelerator.

**Note.** The security of your PeopleSoft system and configuration of load balancers, switches, and reverse proxy servers is beyond the scope of this document. Refer to your PeopleBooks for more information.

REN server access from browser clients is restricted to users who are currently signed in to PeopleSoft software with appropriate REN server permissions. You must enable single sign-in security to obtain REN server access. Permission to access REN server applications is granted on permission lists, which are in turn associated with security roles and user IDs. Clients lacking access permission receive a 403 Forbidden page from the REN server.

**Note.** REN server access requires that single sign-in be enabled.

### *See Also*

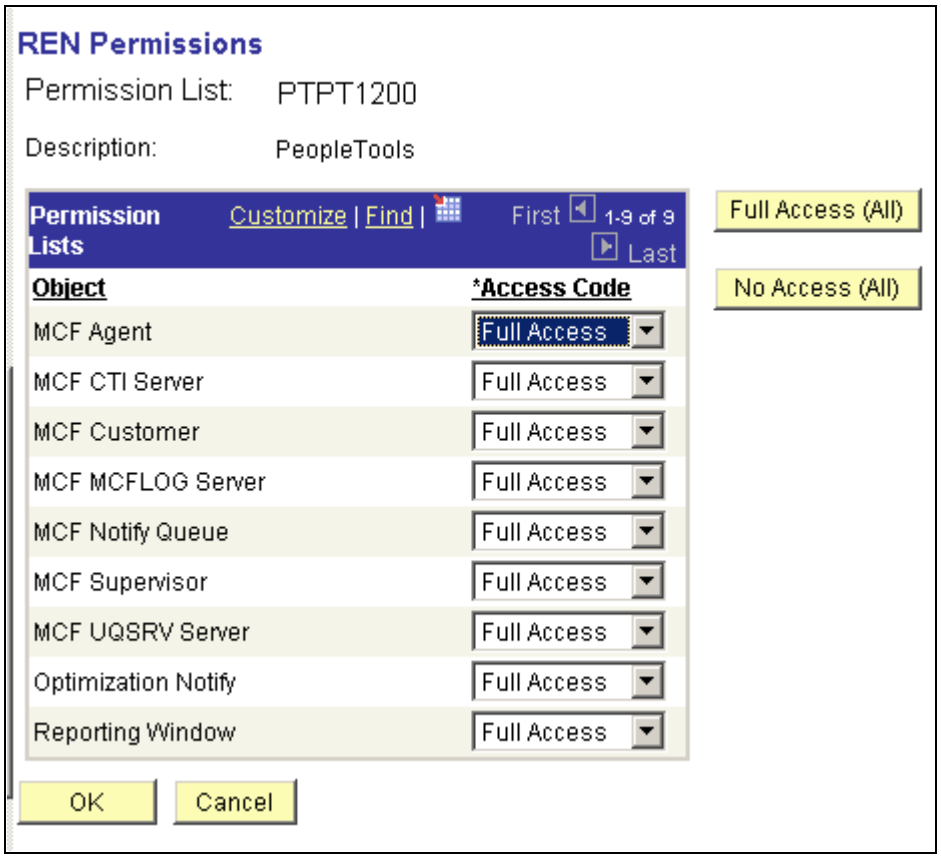
*PeopleTools 8.51 PeopleBook: Security Administration*, "Getting Started with Security Administration"

*PeopleTools 8.51 PeopleBook: System and Server Administration*, "Getting Started with System and Server Administration"

## Defining Permission Lists for REN Server Access

The following REN Permissions page shows the objects and permissions that are defined for permission list PTPT1200. You can create custom permission lists and define access to REN servers.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The REN Permissions page showing the objects and permissions that are defined for a permission list

To define permission lists for REN server access:

1. Select PeopleTools, Security, Permissions & Roles, Permission Lists.
2. On the search page, search for and select your permission list.
3. On the Permission List page, select the PeopleTools tab.
4. Click Realtime Event Notification Permissions.
5. On the REN Permissions page, select your permissions.

   To enable REN server access for roles that are defined with the current permission list, select *Full Access* for each object that is required by the role. For example, users who require access to the MultiChannel Console must have Full Access defined for the MCF Agent object.

   The MultiChannel Console link appears in the universal navigation header for any user with full access permissions defined for the MCF Agent object. However, the user must also be configured as an MCF or CTI agent to access the MultiChannel Console or CTI console.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Note.** To enable access to the Report-to-Window functionality, add WEBLIB_RPT to the Web Libraries page of the permission list, and set Reporting Window to Full Access on the REN Permissions page.

Grant full access to the MCF CTI Server object only on the permission list that is assigned to the CTI server role. No other users should have MCF CTI Server access.

The user ID that is configured to start the Process Scheduler must have full access to the Reporting Window REN permission on at least one permission list for that user ID. If the user ID does not have full access to the Reporting Window, then the pop-up window stays in a status of queued.

### *See Also*

Chapter 3, "Configuring PeopleSoft Computer Telephony Integration," Required Security for PSMCAPI, page 20

# Configuring REN Servers

To configure REN servers, use the REN Server (REN_SERVER_CMP) component.

This section provides an overview of REN server configuration options and discusses how to:

- Configure REN servers and SSL-enabled REN servers.
- Define REN servers.

## Understanding REN Server Configuration Options

Depending on your requirements, choose one of two REN server creation and configuration options:

- To create a single REN server in a particular database using default configuration parameters, create an application server domain using PSADMIN.

  Event Notification is enabled by default in the quick-configure menu. An associated REN server cluster is also created by default.

- To create additional REN servers in a particular database, configure each REN server as required on the REN Server Definition and REN Server Cluster pages.

  Then create the associated application server domains. Event Notification is enabled by default in the quick-configure menu.

When a REN server starts, it looks for configuration information in the database, using the application server domain name and host name as keys. If the associated configuration information exists in the database, the REN server uses it. If no such configuration information exists, the REN server is configured by defaults, which also configure a REN server cluster for each REN server. You can change the default REN server configuration by using the REN Server Configuration pages, but such changes do not take effect until the REN server starts up again.

**Note.** You can create only one REN server per application server domain.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

This section discusses some possible REN server configurations that depend on domain server topology.

### *Simple Configuration: Mycompany.com*

In this configuration, the REN server is on the host machine MachA, the REN server uses the default port number 7180, the domain name server (DNS) addresses the host machine as MachA.mycompany.com, and no SSL or reverse proxy server is involved:

| *Parameter* | *Value* |
|---|---|
| PeopleSoft Pure Internet Architecture Authentication Token Domain | mycompany.com |
| Authentication Domain in REN Server Cluster Configuration | mycompany.com |
| REN Server Cluster Root Path | /psren |
| REN Server Cluster URL | http://MachA:7180 |
| REN Server Browser URL | http://MachA.mycompany.com:7180 |

### *Simple Configuration with SSL-Enabled REN Server: Mycompany.com*

In this configuration, the REN server is on the host machine MachA, the REN server uses the default port number 7143, and DNS addresses the host machine as MachA.mycompany.com. The REN server is SSL-enabled.

| *Parameter* | *Value* |
|---|---|
| PeopleSoft Pure Internet Architecture Authentication Token Domain | mycompany.com |
| Authentication Domain in REN Server Cluster Configuration | mycompany.com |
| REN Server Cluster Root Path | /psren |
| REN Server Cluster URL | https://MachA:7143 |
| REN Server Browser URL | https://MachA.mycompany.com:7143 |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Reverse Proxy Server with Non-SSL Configuration*

This configuration includes a single REN server and a reverse proxy server. The reverse proxy server could be either a dedicated reverse proxy server or a web server with a proxy plug-in configured to redirect both PeopleSoft Pure Internet Architecture and REN server requests. The application server host machine is MachA, and the REN server uses its default port 7180. The reverse proxy server is on MachRPS using port 8080 for HTTP. The DNS server must recognize MachRPS.mycompany.com.

| *Parameter* | *Value* |
|---|---|
| PeopleSoft Pure Internet Architecture Authentication Token Domain | mycompany.com |
| Authentication Domain in REN Server Cluster Configuration | mycompany.com |
| REN Server Cluster Root Path | /psren |
| REN Server Cluster URL | http://MachA:7180 |
| REN Server Cluster Browser URL | http://MachRPS.mycompany.com:8080 |

### *Reverse Proxy Server with SSL Configuration and Secure HTTP*

For SSL, install certificates on the reverse proxy server, set the server to encrypt all communications, and use HTTPS URLs from the browser. In this example, the reverse proxy server uses port 8443 for SSL:

| *Parameter* | *Value* |
|---|---|
| PeopleSoft Pure Internet Architecture Authentication Token Domain | mycompany.com |
| Authentication Domain in REN Server Cluster Configuration | mycompany.com |
| REN Server Cluster Root Path | /psren |
| REN Server Cluster URL | http://MachA:7180<br><br>**Note.** The cluster URL should not be a secure HTTP address if SSL is handled through a reverse proxy server. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Parameter* | *Value* |
| --- | --- |
| REN Server Browser URL | https://MachRPS.mycompany.com:8443<br>**Note.** This is a secure HTTP address (HTTPS). |

**Note.** If you use SSL between the browser and REN server, you must use a reverse proxy server or SSL accelerator.

### *See Also*

*PeopleTools 8.51 PeopleBook: Security Administration*, "Getting Started with Security Administration"

## Page Used to Configure REN Servers

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
| --- | --- | --- | --- |
| REN Server Configuration | REN_SERVER_DET_PG | PeopleTools, REN Server Configuration, REN Server Definition | Define a REN server. |

## Configuring REN Servers and SSL-Enabled REN Servers

Specify REN server configuration parameters based on your network topology and server arrangement.

Define the parameters for REN server configuration in three locations:

- Authentication token domain, set during PeopleSoft Pure Internet Architecture installation or in web profile configuration.
- REN server configuration parameters, specified in an application server domain using PSADMIN.
- REN server parameters, including cluster and browser URLs, set in the PeopleTools REN Server and REN Cluster components.

  Configuration parameters that are set in the REN Server and REN Cluster components override any defaults in PSADMIN.

### *Authentication Domain*

The authentication domain tells PeopleSoft Pure Internet Architecture the internet domain name that browser clients use when accessing PeopleSoft applications across the internet. The token is required to comply with the same-origin security policy that is enforced by most browsers. The domain name that is specified in the REN Server Configuration page must be identical to the domain name that is specified as the authentication token domain during PeopleSoft Pure Internet Architecture installation.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

If authentication domain is not set during PeopleSoft Pure Internet Architecture installation, define the authentication domain in web profile configuration to match the REN server configuration.

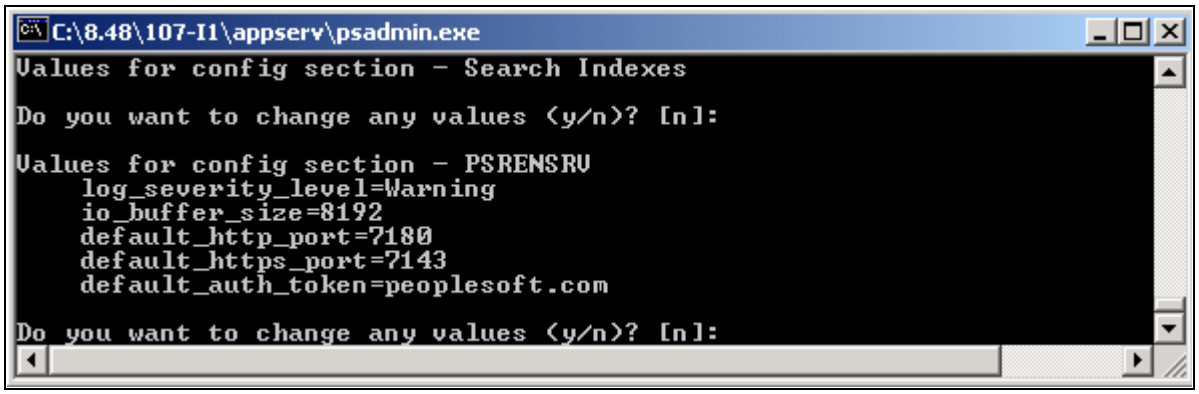
---

**Note.** You must specify the authentication token domain if you access the REN server and the PeopleSoft Pure Internet Architecture web server using different DNS names from the browser client (for example, if they are on different machines).

---

### *Configuring a REN Server and SSL-Enabled REN Server with PSADMIN*

If necessary, you can specify parameters in the PSRENSRV section of the PSADMIN application server domain configuration, as illustrated in the following example:




Configuring PSRENSRV in PSADMIN

Specify parameters as described in the following table:

| *Parameter* | *Default* | *Description* |
|---|---|---|
| log_severity_level | Warning | This is the logging level for the REN server.<br>Select from one of the following log severity levels, from less to more logged data: Error, Warning, Notice, Debug.<br>**Note.** Do not use Debug in a production environment. |
| io_buffer_size | 8192 | This is the TCP buffer size in bytes that is used for serving content. Do not exceed a value of 65536.<br>If the REN server is running on Microsoft Windows, change io_buffer_size to a minimum value of 56000. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Parameter* | *Default* | *Description* |
|---|---|---|
| default_http_port | 7180 | This is the REN server's HTTP port.<br>The default value is *7180.*<br>The default_http_port parameter takes effect only when a REN server starts up for the first time and the database does not already contain configuration information for the REN server.<br>**Note.** After the HTTP port number that is assigned to the REN server has been established in the database, the only way to change it is on the REN Server Definition page. Editing the port number in the psappsrv.cfg file does not overwrite the value that is stored in the database. |
| default_https_port | 7143 | This is the REN server's HTTPS port for SSL-enabled REN server.<br>The default value is *7143.*<br>**Note.** The https port is used only when the REN server is SSL-enabled.<br>The default_https_port parameter is configured in psappsrv.cfg and is used when a SSL-enabled REN server starts up for the first time.<br>**Note.** To change the default https port, use the REN Server Definition page. Changing the SSL Port requires the reboot of the REN server. |
| default_auth_token | example.com | This is the fully qualified domain name of the application server.<br>This value should match the value of the web server's authentication domain.<br>The default_auth_token parameter takes effect only when a REN server starts up for the first time and the database does not already contain configuration information for the REN server.<br>When configuring the REN server parameters through PSADMIN, do not place a period or dot (.) before the default_auth_token value. For example, the parameter should read default_auth_token=example.com |

After specifying REN server configuration parameters, be sure to specify Y (Yes) when asked if you want event notification configured and MCF server configured. Boot this domain from the Domain Administration menu.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Note.** Use PeopleSoft Pure Internet Architecture REN server definition and configuration pages to modify configuration parameters whenever possible. REN server configuration parameters that you make using PSADMIN are written to the psappsrv.cfg file in the application server directory. REN server configuration values that are found in the database override any values that are found in psappsrv.cfg.

Use static IP addresses for your web servers. If you use dynamic IP addresses (DHCP), ensure that the domain name server (DNS) can map fully qualified domain names to the dynamic IP addresses.

If you are using Microsoft Internet Explorer internet security zones, include both the web server and REN server addresses in the same security zone; alternatively, exclude both addresses from security zones.

### *Socket Binding*

The REN server listens on the port that is defined in the REN Server Definition page, which is by default *7180*. However, the host name to which the REN server binds is determined by information in the psrenconfig.txt file for each application server domain. If the host machine contains multiple network interface cards (NICs), then the REN server binds by default to only one NIC, which is given by `uname()` on Unix, or `GetComputerName()` on Microsoft Windows.

To bind a REN server to a specific NIC, manually edit psrenconfig.txt for the appropriate application server domain, changing both set address and set hostname to the IP address and locally-known host name of the NIC. For example:

```
set address   192.168.10.1
set hostname  hostsrv.example.com
```

**Note.** If you enter an invalid IP address in the psreconfig.txt file, the REN server may not start correctly. Check the REN server log for error messages that identify the issue.

### *Configuring TCP_NODELAY*

The parameter, TCP_NODELAY in psrenconfig.txt controls whether to disable the TCP Nagle algorithm on the TCP packets sent by the REN server. Two instances of TCP_NODELAY are available in psrenconfig.txt. TCP_NODELAY in the nssock section is used by non-SSL REN servers, and the instance in the nsopenssl section is used by SSL-enabled REN servers. The TCP Nagle algorithm is generally enabled by default and inserts a short delay before sending small TCP packets. This helps prevent network overload.

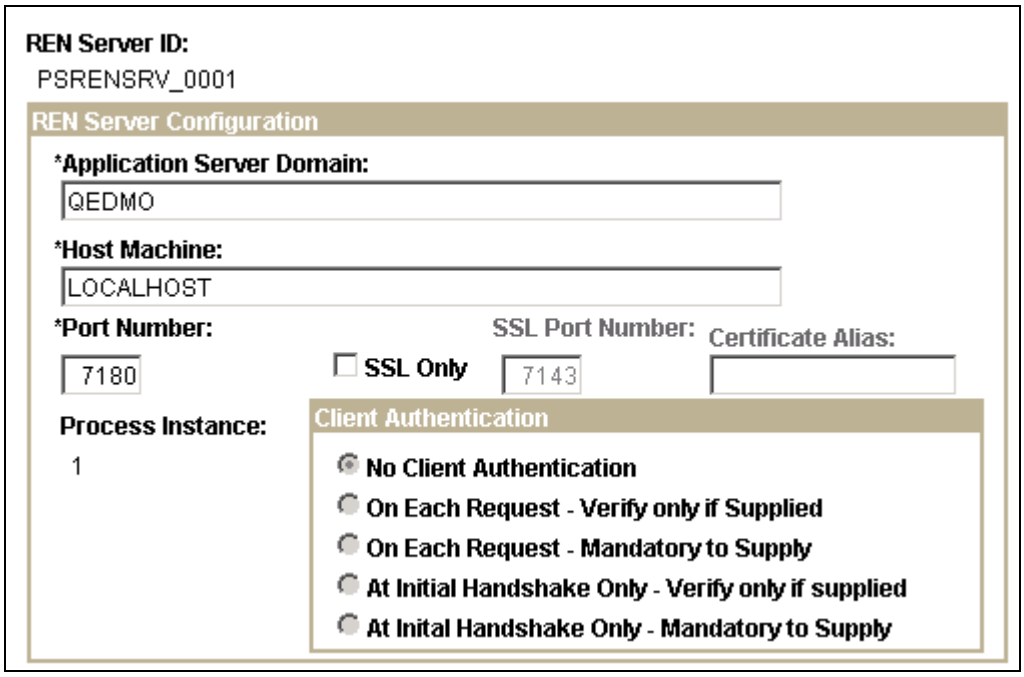
If TCP_NODELAY is set to 0, the TCP algorithm acts normally. This is the recommended configuration for most applications. However, for certain CTI applications, this parameter must be set to 1 to improve performance. If TCP_NODELAY is set to 1, the TCP Nagle algorithm is disabled on operating system platforms that support disabling this feature.

## Defining REN Servers

Access the REN Server Configuration page using the following navigation path:

PeopleTools, REN Server Configuration, REN Server Definition

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 




The REN Server Configuration page displaying the REN server ID and the Process Instance. It has the following editable options: Application Server Domain, Host Machine, Port Number, SSL Only check box, SSL Port Number, Certificate Alias, and the various Client Authentication options to choose from.

**Application Server Domain** — Enter the application server domain that is serving this REN server.

**Host Machine** — Enter the name of the host machine on which the specified application server domain runs.

This entry requires the host machine name, not its DNS name. However, the host machine name may need to be fully qualified, for example, machineA.example.com. On a Unix machine, determine the host name by running `uname -a`. On a Microsoft Windows machine, determine the host name by running `hostname` at a command prompt.

**Port Number** — Enter the HTTP port number on which this REN server is addressed.

Change the HTTP port value if multiple REN servers are running on the same host machine to avoid port conflicts.

**SSL Only** — Select to enable SSL on REN server.

---

**Note.** If this option is selected, you must enter the SSL port.

---

**SSL Port** — Enter the HTTPS port number on which this SSL-enabled REN server is addressed.

**Process Instance** — Reserved for future use.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Certificate Alias** Select a certificate alias to be used as a server certificate by the SSL-enabled REN server.

**Note.** The certificate alias is stored in the PSKEYDB, PSCERTDB, and PSREN records.

**Client Authentication** Select to determine the level of client authentication.

**Note.** If the browser is configured for client authentication pop-up or the browser has more than one certificate configured, the SSL session ends if the user fails to provide the certificate within three heartbeats. To avoid such a session time-out, the user must either accept the client certificate within a heartbeat or increase the session time-out value in psrenconfig.txt.

The following table shows the client authentication values:

| *Parameter* | *Flag Value* | *Description* |
|---|---|---|
| No Client Authentication | 0 | Client authentication is disabled. |
| On each Request-Verify only if Supplied | 1 | Client authentication is enabled<br>The server sends a client certificate request to the client. Verification happens only if the certificate is provided. If the verification process fails, the TLS/SSL handshake is immediately terminated. If the client does not return any certificate, SSL communication still continues |
| On Each Request-Mandatory to Supply | 3 | Client authentication is enabled and mandates that the client provide the certificate.<br>If the client does not return a certificate, the TLS/SSL handshake is immediately terminated with a handshake failure alert. If the client returns a certificate, it is verified. The communication fails if the verification fails. |
| At Initial handshake Only-Verify only if Supplied | 5 | Client authentication is enabled and requests a client certificate on the initial TLS/SSL handshake only.<br>Verification happens only if the certificate is provided. If the client does not provide any certificate, SSL communication still continues. If verification fails, the TLS/SSL handshake is immediately terminated. |
| At Initial handshake Only-Mandatory to Supply | 7 | Client authentication is enabled and mandates that the client provide the certificate only in initial TLS/SSL handshake. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Configuring REN Server and SSL-Enabled REN Server Clusters

A cluster is typically a collection of REN servers among which the session information is replicated. You cannot add both SSL and non-SSL servers in a single cluster.

To configure REN server clusters, use the REN Cluster (REN_CLUSTER_CMP) component.

This section discusses how to:

- Cluster REN servers and SSL-enabled REN servers.
- Specify REN server ownership.
- Specify REN server cluster members.

REN server clusters address failover and scalability.

## Pages Used to Configure REN Server and SSL-Enabled REN Server Clusters

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| REN Server Cluster | REN_CLUSTER_PG | PeopleTools, REN Server Configuration, REN Server Cluster, REN Server Cluster | Define a REN server cluster. |
| REN Server Cluster Owner | REN_OWNER_PG | PeopleTools, REN Server Configuration, REN Server Cluster, REN Server Cluster Owner | Specify REN server cluster ownership. |
| REN Server Cluster Members | REN_CLUST_RSERV_PG | PeopleTools, REN Server Configuration, REN Server Cluster, REN Server Cluster Members | Specify REN server cluster members. |

## Defining a REN Server Cluster

A REN server serves requests only if it is a part of the cluster. If the REN server is SSL-enabled:

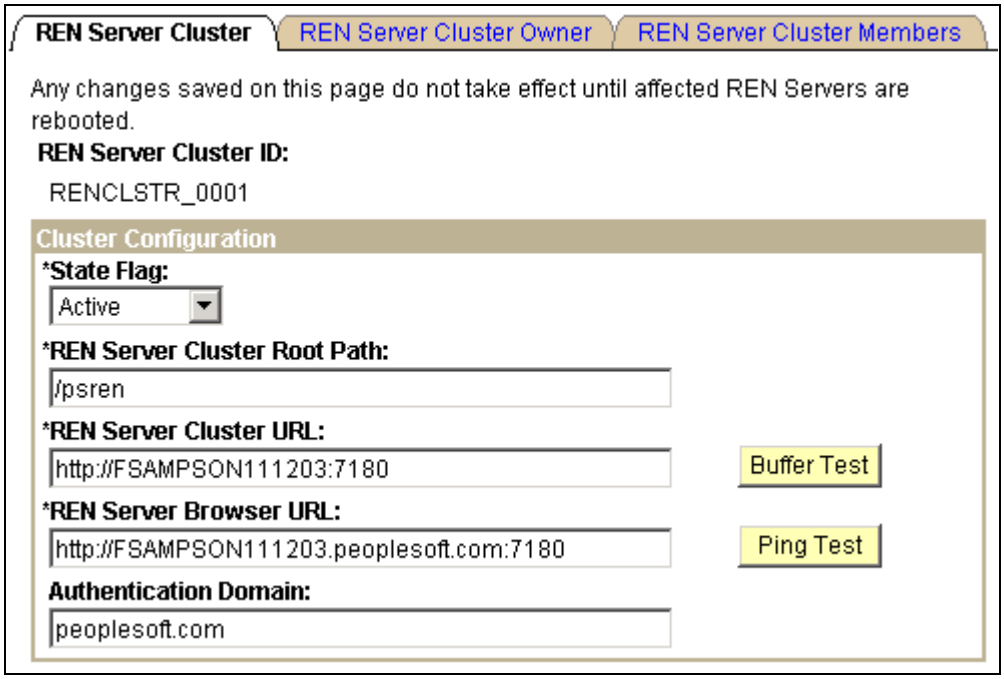
- All the member REN servers must be SSL-enabled REN servers.
- All the member REN servers must use the same server certificate.
- Both REN Server Cluster URL and REN Server Browser URL must start with HTTPS and use the HTTPS port.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Note.** When the administrator changes the REN server to be in SSL mode, he or she must also ensure that the REN server is a member of SSL clusters only. In any given REN cluster, all REN servers that are members must be either SSL-only servers or non–SSL servers. For SSL-enabled REN servers, use SSL-enabled PeopleSoft Pure Internet Architecture.

Access the REN Server Cluster page using the following navigation path:

PeopleTools, REN Server Configuration, REN Server Cluster, REN Server Cluster




The REN Server Cluster page showing the REN server cluster ID and having the following editable options: State Flag, REN Server Cluster Root Path, REN Server Cluster URL, REN Server Browser URL, and the Authentication Domain

By default, if you start a REN server from PSADMIN without configuring a REN server cluster, a cluster is created with a cluster ID RENCLSTR_000n

**State Flag**

Select *Active* or *Inactive.*

This field determines whether the cluster can receive new client requests. For scalability, configure multiple REN server clusters with the same ownership and set them to active status. Then the reporting window and customer chat applications will direct new client requests to a randomly chosen active REN server cluster. If all clusters are inactive, the client receives an error message.

If the cluster supports MCF servers, current chat sessions continue even after a cluster is inactive. But the MCF system does not route any additional requests to an inactive cluster.

Inactivate a cluster before deleting the cluster, or before removing a member REN server from the cluster. You can inactivate a REN server cluster without deleting the cluster.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**REN Server Cluster Root Path**

The default REN server cluster root path is /psren. Change this as required so that multiple REN server clusters are addressable through a single reverse proxy server.

Changes to the root path should also be reflected in the URL mapping of any reverse proxy server.

**REN Server Cluster URL**

The REN server cluster URL is the address that is used to reach the REN server cluster internally.

This is the URL that is used by internal processes. If the MCF cluster is served by a REN server cluster, the cluster URL is that of the switch or load balancer in front of the clustered REN servers. The cluster URL must be unique for each cluster. No two clusters can address the same cluster URL. Specify the cluster URL in the form <http://<DNS_machine_ name>:<port>, where:

- <DNS_machine_name> is the server machine name that is recognized by your DNS.
- <port> is the REN server port number; the default value is 7180.

  This port number is the REN server port number or the port number of a proxy server, load balancer, or other front end.
- The protocol must be HTTP if the REN server is non-SSL; for an SSL-enabled REN server, the protocol must be HTTPS.

**Buffer Test**

Click Buffer Test to initiate a test of the REN servers' ability to break up and send a large file using multiple internal buffers.

The buffer test bypasses REN server security, and does not depend on specified domain names (authentication domain), so you can use it to verify that the REN server is running on the network.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**REN Server Browser URL**

The REN server browser URL is the address that is used by external clients and by agent chat to reach the application that is served by this REN server cluster.

The browser URL may be different from the cluster URL, which should not have to go through any firewall, reverse proxy server, or other outward-facing security barrier. If the REN server is reached through a load balancer, switch, or reverse proxy server, specify the fully qualified URL of that device as accessed from the user's browser. The URL must be the address of the gateway machine (proxy server, load balancer, or SSL accelerator). Specify the address in the form http: or https://<DNS_machine_ name>.<domain_name>:<port>, where:

- <DNS_machine_name> is the server machine name that is recognized by your DNS.
- <domain_name> is the fully qualified domain name that is recognized by your DNS.
- <port> is the REN server port number; the default value is 7180.

  This port number is the REN server port number or the port number of a proxy server, load balancer, or other front end.

**Note.** If the REN server is SSL-enabled, the browser URL must be HTTPS.

**Ping Test**

Click to initiate a test of the REN server that is specified in the browser URL fields. Failure may indicate that a URL or authentication domain is incorrectly specified, the REN server is not running, or single sign-in is not implemented.

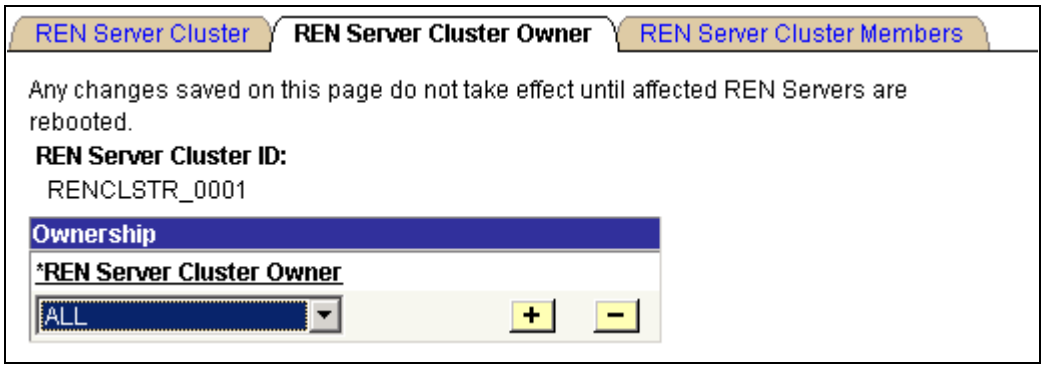
**Authentication Domain**

Enter the authentication domain. This must be the same as the authentication domain that is specified in the PeopleSoft Pure Internet Architecture installation or in the web profile configuration.

## Specifying REN Server Ownership

Access the REN Server Cluster Owner page using the following navigation path:

PeopleTools, REN Server Configuration, REN Server Cluster, REN Server Cluster Owner




The REN Server Cluster Owner page showing the REN server cluster ID and having the editable REN Server Cluster Owner editable field

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**REN Server Cluster Owner**

Select the owner of this REN server cluster from the drop-down list box. Select from the following values:

- *All*
- *MCF*
- *Optimization*
- *Reporting*

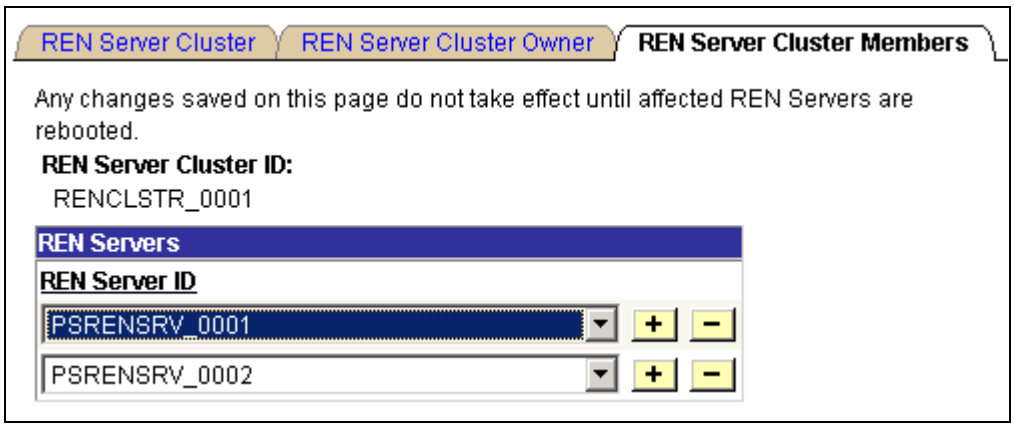
Specifying an owner for a REN server cluster limits client access to that cluster. This is useful to ensure performance under load.

Specifying an owner for a REN server cluster also supports security. For example, an MCF cluster can be created only on a REN server cluster that is owned by MCF or ALL.

## Specifying REN Server Cluster Members

Access the REN Server Cluster Members page using the following navigation path:

PeopleTools, REN Server Configuration, REN Server Cluster, REN Server Cluster Members




The REN Server Cluster Members page showing the REN server cluster ID and having the REN Server ID editable field

**REN Server ID**

Select a REN server from the drop-down list box.

Each REN server can belong to only one REN server cluster.

# Configuring a Reverse Proxy Server with a REN Server

This section provides an overview of reverse proxy server (RPS) configuration and provides examples.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Understanding RPS Configuration

Production PeopleSoft installations may configure the REN server behind an RPS. The RPS isolates the REN server and other web servers from the open internet, provides SSL session handling, and presents a single-server origin to outside clients. PeopleSoft customers may put REN servers and PeopleSoft Pure Internet Architecture web servers behind one RPS, or just REN servers.

These examples assume that:

- You have installed PeopleTools 8.48 or higher on both host machines.
- You have configured a web server using the default parameters on the first host machine.
- You have configured a REN server using the default parameters on the first host machine.

See Chapter 5, "Configuring REN Servers," Reverse Proxy Server with SSL Configuration and Secure HTTP, page 73.

## Example: Configuring a WebLogic RPS for a REN Server on Another Host Machine

This example presents one possible configuration for a REN server running on one host machine and installing an RPS to run on a second host machine, using Oracle WebLogic . The RPS redirects clients to both a REN server and to the PeopleSoft Pure Internet Architecture web server.

To configure an RPS for a REN server on another host machine:

1. Install a new web server domain on the second machine.

   Name the domain *rps.*

   Configure the following values:

   - AppServer Name: <*application_server_machine_name*>
   - JSL Port: *9999*

     The RPS will not make Jolt connections.
   - HTTP Port: *8080*
   - HTTPS Port: *8443*

2. Start the new web server.

   Navigate to <PIA_HOME>webservrps, and run `startPIA.cmd`.

3. Sign in to the WebLogic Server Administrative Console for the *rps* web server.

   Access the WebLogic Server Administrative Console at http://<webserver>:<port>/console (for example, http://localhost:8080/console).

   When prompted for a user name and password, specify the WebLogic system ID and password. If you've followed the default WebLogic Server install, the ID and password are *system* and *password.*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

4. Using the console's hierarchical navigation, navigate to rps, Deployments, Applications, PeopleSoft. Select the Targets tab.

   Clear the PIA option.

   Click Apply.

5. Using the console's hierarchical navigation, navigate to rps, Deployments, Web Application Modules, HttpProxyServlet. Select the Targets tab. Select the PIA option. Click Apply.

6. For better web server performance, navigate to rps, Servers, PIA. Select the Protocols tab, select the HTTP tab, and set both Duration and HTTPS Durationto *120* secs.

7. Stop the *rps* web server.

   Navigate to <PIA_HOME>webservrps and run `stopPIA.cmd`.

8. Configure RPS parameters for the *rps* server.

   Locate the file web.xml at PIA_HOME/webserv/rps/applications/HttpProxyServlet/WEB-INF.

   Edit web.xml in a text editor, changing the WebLogic port and WebLogic host from *8080* to *80* (the value *8080* is a default value that is derived during installation of the domain *rps).* For example:

```
<init-param>
 <param-name>WebLogicPort</param-name>
 <param-value>80</param-value>
 <description>HTTP listen port of WebLogic PIA/PORTAL server.</description>
</init-param>
```

   To specify the associated REN server, (which is on another machine), edit web.xml, changing the REN server host machine, port, and root URL from their default RPS values. For example:

```
<init-param>
 <param-name>WebLogicHost</param-name>
 <param-value>MACHINE_2</param-value>
 <description>Hostname of REN server.</description>
</init-param>
<init-param>
 <param-name>WebLogicPort</param-name>
 <param-value>7180</param-value>
 <description>Listen port of REN server.</description>
</init-param>
```

   Another example is:

```
<servlet-mapping>
 <servlet-name>RENHttpProxyServlet</servlet-name>
 <url-pattern>/psren/*</url-pattern>
</servlet-mapping>
```

9. Reboot the RPS web server.

   Navigate to <PIA_HOME>\webserv\rps, and run `startPIA.cmd`.

10. (Optional) Configure and enable SSL on the RPS machine.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Note.** When using Apache 1.3.x or 2.0.x RPS, you must configure the kn_response_flush_override and the flush_rps_buffer_size_for_knjs parameters in the psrenconfig.txt file. If you are using Apache 1.3.x, set both of these parameters to *4096.* If you are using Apache 2.0.x, set both parameters to *8192.* Apache needs both parameters present with the same buffer size. The kn_response_flush_override parameter flushes a message, while the flush_rps_buffer_size_for_knjs parameter flushes the stay-alive.

**Note.** Using WebLogic as a reverse proxy server is not recommended for a production system.

## Configuring Apache-based Reverse Proxy Servers for a REN Server

Apache-based proxy servers vary widely in configurations, here we present an example configuration. The configuration files for your environment may be quite different.

To proxy for RenServer, find and edit the httpd.conf configuration file. Make the following modifications to the file:

1. Move the line LoadModule proxy_module modules/ApacheProxyModule.dll to the bottom of the file.
2. Comment out the line AddModule mod_proxy.c.
3. Add the following five lines after LoadModule proxy_module:

```
<IfModule mod_proxy.c>
  ProxyRequests Off
  ProxyPass /psren http://machine:7180/psren
  ProxyPassReverse /psren http://machine:7180/psren
</IfModule>
```

4. Reboot your webserver and reverse proxy server.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

Chapter 6

# Configuring PeopleSoft MCF Servers and Clusters

This chapter provides an overview of PeopleSoft MultiChannel Framework (MCF) server and cluster architecture and discusses how to configure MCF clusters.

## Understanding PeopleSoft MCF Server and Cluster Architecture

This section discusses:

- PeopleSoft MCF server configuration.
- PeopleSoft MCF cluster architecture.
- Queue server and queue server failover.
- Logical queues and physical queues.
- MCF log server and log server failover.
- Queue server scalability.
- Recommended configurations.

### PeopleSoft MCF Server Configuration

PeopleSoft MCF depends on processes that are configured and booted as part of an application server domain. Configure MCF processes (servers) through PSADMIN, along with other processes in each application server domain.

**Note.** The REN server process can be used by applications that are separate from the queue server and MCF log processes. In this case, you can configure the application server domain for event notification without creating the MCF servers.

After considering performance and failover issues, the MCF system administrator provides configuration information that describes the arrangement of queues, domains, queue server processes, REN server processes, MCF processes, and URL addresses.

REN server processes are configured on PeopleSoft Pure Internet Architecture pages before or after being initiated in an application server domain. Each queue server process is uniquely identified in the system by the combination of the machine name, domain subdirectory name, and process identifier. MCF log processes use the same queue-server identification scheme.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

***See Also***

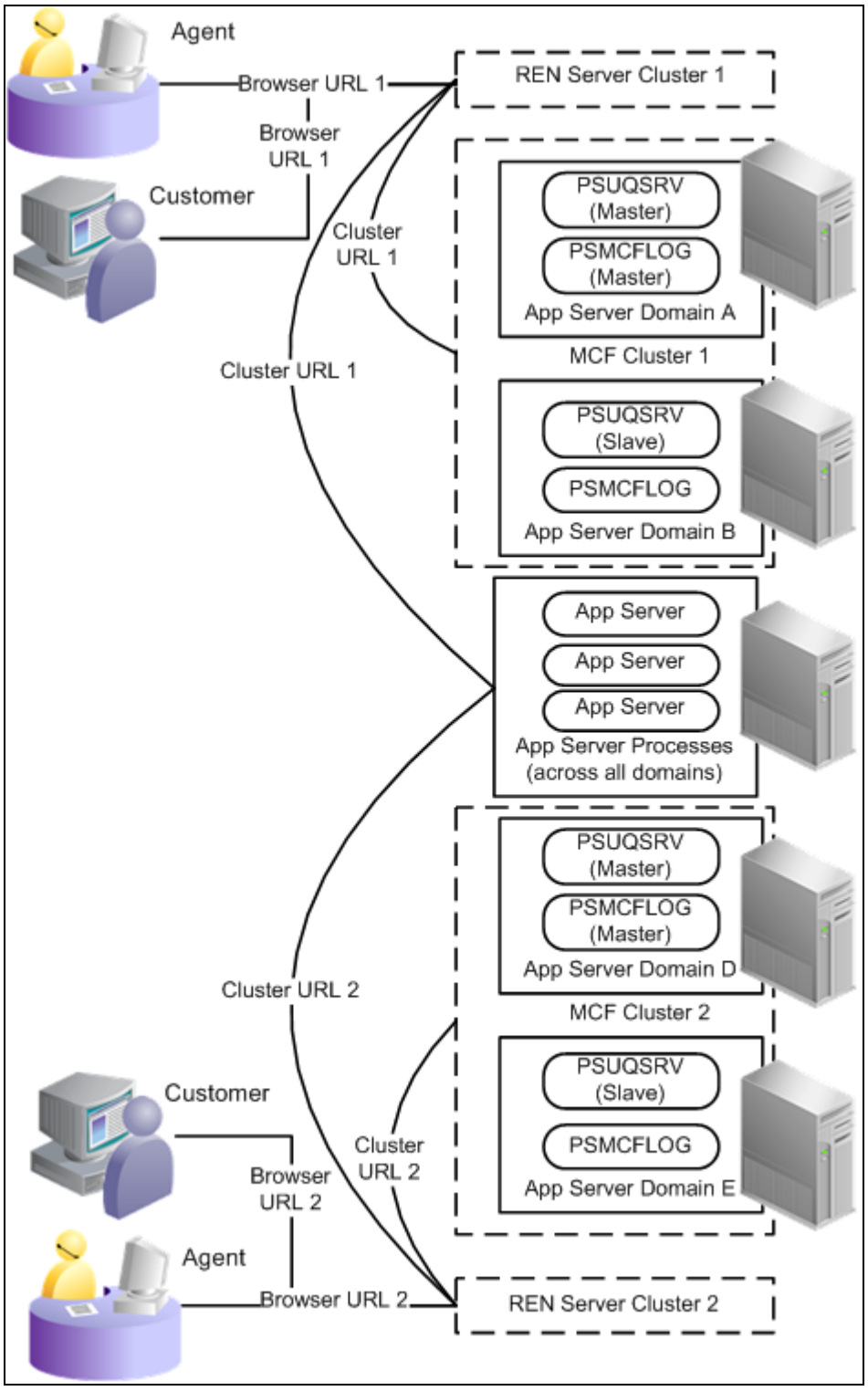
Chapter 5, "Configuring REN Servers," page 65

## PeopleSoft MCF Cluster Architecture

In a PeopleSoft system, all application server processes, including MCF servers and REN servers, belong to an application server domain. Each domain can have only one REN server process (PSRENSRV), one queue server process (PSUQSRV), and one MCF log server process (PSMCFLOG). Domains can be redundantly clustered to provide failover. Logical queues can be serviced by multiple clusters for scalability. Support for scalability and failover is integrated into the configuration process.

The following diagram illustrates MCF cluster architecture:

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




PeopleSoft MultiChannel Framework cluster architecture

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Queue Server and Queue Server Failover

The queue server is a server process in the PeopleSoft application domain that routes email, chat, and generic tasks to the agent based on the agent properties, such as state and skill set, and task routing properties, such as priority, language, and cost.

The queue server process (PSUQSRV) is a Tuxedo-managed server with a standard PeopleSoft database connection. Each queue server process is the central routing point for one or more physical queues. The queue server maintains state information for work requests, work in progress, agent availability, and agent workload. Queue server state is written to database records, except for the assignment of chat to agents.

The queue server can recover from a crash because most of its state is written to the database. When a queue server reboots, it checks the database and loads state information for open work tasks. The queue server connects to its REN server and issues restart queries to each console so that it can rebuild agent assignment information, which may have changed while the queue server was down.

Although a single queue server can recover state after recovering from software failure, this does not guard against hardware failure. Multiple queue server processes running on multiple host machines and configured in a cluster provide failover for hardware failure. Unlike the REN server cluster, the clustered queue servers operate as one master and many slaves. The master handles all routing decisions, while slave processes monitor the master and step in only if it fails. Any rebooted queue server rejoins the cluster in a slave role. Any slave that is promoted to master loads state from the database and issues queries to consoles as if it were the only process in the cluster.

Each queue server process follows a fixed procedure to ensure that the cluster has at most one functioning master. Database locks eliminate possible race conditions, and the master periodically writes a timestamp to indicate its health. The masterinterval parameter controls the frequency at which the master process must update the timestamp in the cluster table. The masterinterval parameter corresponds to the maximum time after a master queue server fails before another queue server process takes over. Minimizing this value provides rapid failover response time but also requires frequent database updates.

See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Tuning Cluster Parameters, page 108.

Each queue server must be part of an MCF cluster, and each MCF cluster must include at least one queue server. An MCF cluster of only one queue server provides no redundancy against hardware failure.

Create a queue server that starts when an application domain is started by selecting MCF servers from the quick-configure menu during application domain configuration.

In summary, configure queue servers to provide hardware failover. Each queue server is part of an MCF cluster. To support hardware failover, distribute master and slave queue servers over multiple hosts. Every queue server in an MCF cluster communicates with the same REN server ID. Therefore, REN server failover is also crucial.

The first queue server that places a valid master entry for itself in the cluster table becomes the master queue server. In most cases, the master queue server is the first queue server started. No configuration parameter exists to designate master or slave queue server within a cluster.

After an MCF cluster's master queue server is established, all other cluster members become slave queue servers. If the slave queue servers within a cluster detect a failure of the master queue server, the remaining slave servers compete to become the master queue server. If the master queue server reboots before a slave takes over, the master queue server also competes. No configuration parameter exists to designate priority among slave servers.

See Chapter 5, "Configuring REN Servers," page 65.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Logical Queues and Physical Queues

PeopleSoft MultiChannel Framework enables the configuration of both logical and physical queues.

A logical queue is an application-level queue that receives work requests (tasks) relating to an application area, such as chat requests regarding sales information, and routes them to agents that are capable of handling the work. For example, you might configure a logical queue called SALES for sales inquiries and another called SUPPORT for support issues.

Logical queues can be partitioned into physical queues for scalability. A physical queue is managed by a single MCF cluster. For scalability, the tasks that are enqueued on a logical queue are distributed by the framework among all available physical queues. For example, the SALES queue could be serviced by four MCF clusters across four physical queues: SALES1, SALES2, SALES3, and SALES4.

Each agent can be assigned to only one physical queue within each logical queue. Each agent can be assigned to multiple logical queues.

## PeopleSoft MCF Log Server and Log Server Failover

The MCF log server (PSMCFLOG) is a Tuxedo-managed server that is similar to the queue server. Each MCF log server receives events that are sent by a REN server and is responsible for writing MCF events to the database.

The MCF log server logs events to PS_MCFUQEVENTLOG. By default, the log server does not log periodic state information broadcasts from the queue server to the MultiChannel Console. If you need to log these events, configure logging on the Cluster Tuning page. You can also configure the log server to log the contents of chat sessions. Chat session logging is deactivated by default. Logged chats are stored in PS_MCFCHATLOG.

If the MCF log server crashes, it resumes functioning immediately after restarting. When the first slave log server detects a failed master, it takes over as the master log server for the cluster. The new master log server again receives all base topics, but it does not log chat sessions that started or continued during the time that the original master log server failed. The new master log server does not log per-agent events for agents that were signed in at, or during, the time of the failure.

An MCF log server is created along with a queue server when you enable MCF servers during application server domain configuration. No specific log server configuration is available during domain configuration.

## Queue Server Scalability

PeopleSoft MultiChannel Framework is scalable to support large-capacity call centers or other large organizations. The basic strategy is to divide the workload by spreading it over several MCF clusters. This is accomplished by creating multiple physical queues for each logical queue and spreading the management responsibility for each physical queue to separate queue server processes, preferably on multiple host machines. This technique should not be confused with failover protection, which also adds processes and machines. In failover, the added processes are clustered together and do not provide performance improvement.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Organize applications using PeopleSoft MultiChannel Framework around logical queues (for example, SALES queue and SUPPORT queue). Incoming work tasks are sent to a logical queue. PeopleSoft MultiChannel Framework then assigns the task to one of the corresponding physical queues. This assignment is random across the queues. The load across the servers is balanced by servicing only one physical queue per logical queue by single MCF cluster.

For example, a logical SUPPORT queue might be split into physical SUPPORT1 and SUPPORT2 queues such that work requests are randomly distributed between the two physical queues. Half the agents receive from one queue and half from the other. This splits the workload evenly between the two queue server processes, while still presenting one logical SUPPORT queue to the application.

## Recommended Configurations

Consider the following configuration options to ensure maximum reliability and scalability of your PeopleSoft MultiChannel Framework installation:

- Configure multiple MCF servers in a cluster across multiple host machines.

  This provides protection against single-point failures.

  Each MCF cluster requires a REN server cluster. Configuring multiple REN server clusters is functionally the same as configuring multiple MCF clusters for scalability. Inside a REN server cluster, configuring multiple REN servers is functionally the same as multiple queue servers for failover.

  See Chapter 6, "Configuring PeopleSoft MCF Servers and Clusters," Configuring PeopleSoft MCF Clusters, page 95.

- Use REN server clusters only for failover.

  REN server clusters do not enhance performance.

- Split logical queues into more than one physical queue if more work is required on that queue than a single process or machine can handle.

- If an application server domain is likely to be restarted regularly for reasons that are not related to PeopleSoft MultiChannel Framework, configure PeopleSoft MultiChannel Framework in a separate domain.

  Regular restarting of MCF servers affects performance because the MCF servers must recover state when they are recycled or when a slave takes over from a master server.

# Configuring PeopleSoft MCF Clusters

An MCF cluster is a group of multiple MCF-enabled application server domains in which all queue servers and log servers communicate with the same REN server cluster. Only one queue server and one log server in an MCF cluster are active at any one time. These are called the masters. The rest are dormant and redundant, and are called the slaves. If a master drops out of the cluster for any reason, the slaves elect a new master to take its place.

This section provides an overview of MCF cluster configuration and discusses how to configure MCF clusters.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Understanding PeopleSoft MCF Cluster Configuration

Each MCF cluster includes a minimum of one queue server and one log server communicating with one REN server. An MCF cluster is typically identified by the ID of the REN server cluster serving it. No configuration limit is placed on the maximum number of queue servers in an MCF cluster.

In MCF architecture, a chat client initiates contact with an agent through the designated external (browser) URL for a REN server. This URL can point to a load balancer, switch, reverse proxy server, or other hardware or software directing requests through a firewall. The external URL is also known as the browser URL because it supports the MCF browser windows (agent chat, customer chat, and MultiChannel Console).

For communication of queue servers, log servers, and application servers with REN servers, for example, handling email requests behind a firewall, you can use an internal URL. Specify both internal (cluster) and external (browser) REN server URLs during MCF cluster configuration.

---

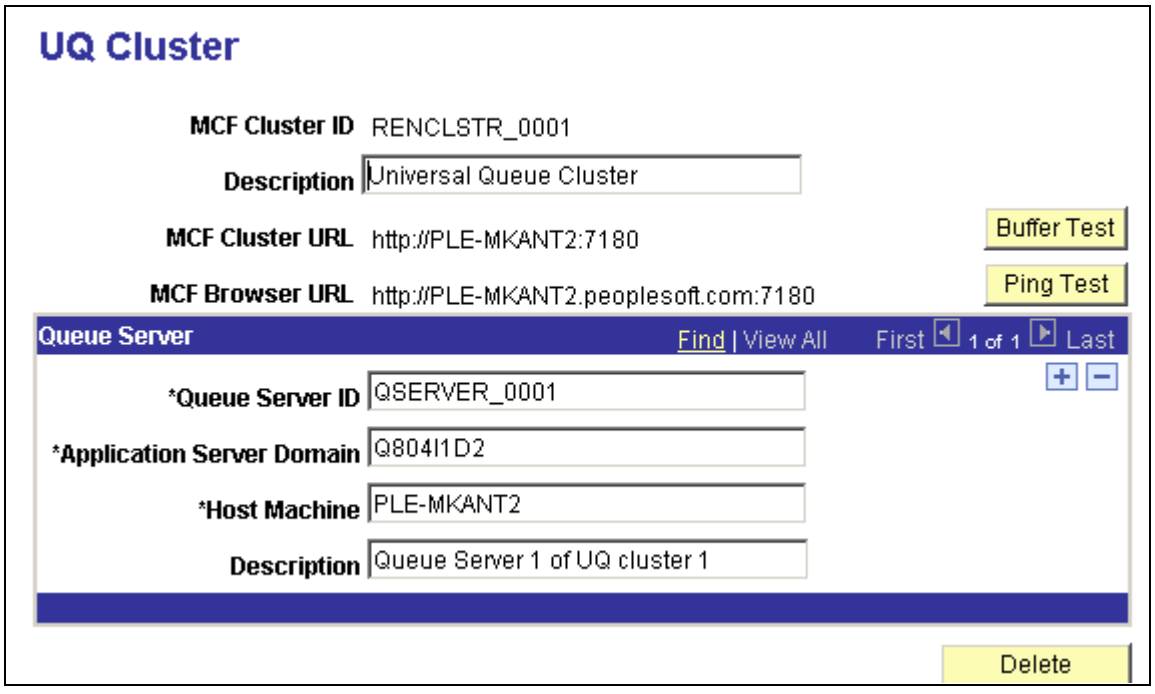
**Note.** If no security is implemented, the browser and cluster URLs may be the same.

---

## Configuring PeopleSoft MCF Clusters

Access the UQ Cluster page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Cluster (MCF_UQCLUSTER)




The UQ Cluster page showing the MCF Cluster ID and having the following editable fields: Description, MCF Cluster URL, MCF Browser URL, Queue Server ID, Application Server Domain, Host Machine, and Description

**MCF Cluster ID** Displays the MCF cluster ID.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**MCF Cluster URL** Displays the URL for the REN server that serves this cluster. This is the URL that is used by internal processes. If the MCF cluster is served by a REN server cluster, the cluster URL is that of the switch or load balancer in front of the clustered REN servers.

**Buffer Test** Click to initiate a test of the REN server's ability to break up and send a large file using multiple internal buffers.

The buffer test does not depend on specified domain names, so you can use it to verify that RENSRV is running on the network.

**MCF Browser URL** Displays the URL for a REN server cluster that serves this MCF cluster for external clients and for agent chat. The browser URL may be different from the cluster URL, which should not have to go through any firewall, reverse proxy server, or other outward-facing security barrier.

**Ping Test** Click to initiate a test of the REN server that is specified in the URL fields. Failure may indicate that a URL is incorrectly specified.

**Delete** Click to delete the entire MCF cluster. No active agents or tasks should be on the cluster.

---

**Note.** If the cluster's queue server configuration is changed, changes to the actual application server domains must be made manually using PSADMIN. For example, if a cluster member (queue server) is removed, the affected application server domain must be shut down and reconfigured (set the MCF Servers field to *No*) using PSADMIN. If the cluster URL is changed, all associated queue server domains must be shut down and rebooted.

---

### *Queue Server*

An MCF cluster can consist of a primary queue server and any number of backup servers.

Each cluster requires a minimum of one queue server. The primary queue server is the first queue server started, and the remaining queue servers are backups. If the primary queue server fails, the system determines the subsequent primary queue server among the backups.

You can add a queue server to a cluster by adding a new row. Before removing a queue server, ensure that it is not the master, and then shut down its domain. Then click Delete (the minus sign).

If a domain is started with a queue server that does not belong to a cluster, the universal queue server and MCF log server poll the MCF configuration tables indefinitely until the queue server is assigned to a cluster.

**Queue Server ID** Enter a unique identifier for each queue server to identify its entries in the database control tables.

The log server process that is paired with this queue server uses this same ID to identify its entry in the log cluster table.

**Application Server Domain** Enter the application server domain of which this queue server is a member.

**Host Machine** Enter the name of the application server host machine.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 7

# Configuring PeopleSoft MCF Queues and Tasks

This chapter discusses how to:

- Define queues.
- Configure tasks.
- View the cluster summary.
- Tune cluster parameters.
- Notify clusters of changed parameters.

***See Also***

*PeopleTools 8.51 PeopleBook: PeopleCode API Reference,* "Universal Queue Classes"

## Defining Queues

To define queues, use the MCF Queue (MCF_Q_CONFIG_CMP) component.

This section discusses how to:

- Define queues.
- Define chat responses.
- Define static push URLs.

### Pages Used to Define Queues

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Queues | MCF_QUEUE_PG | PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Queue, Queue | Define a queue. |

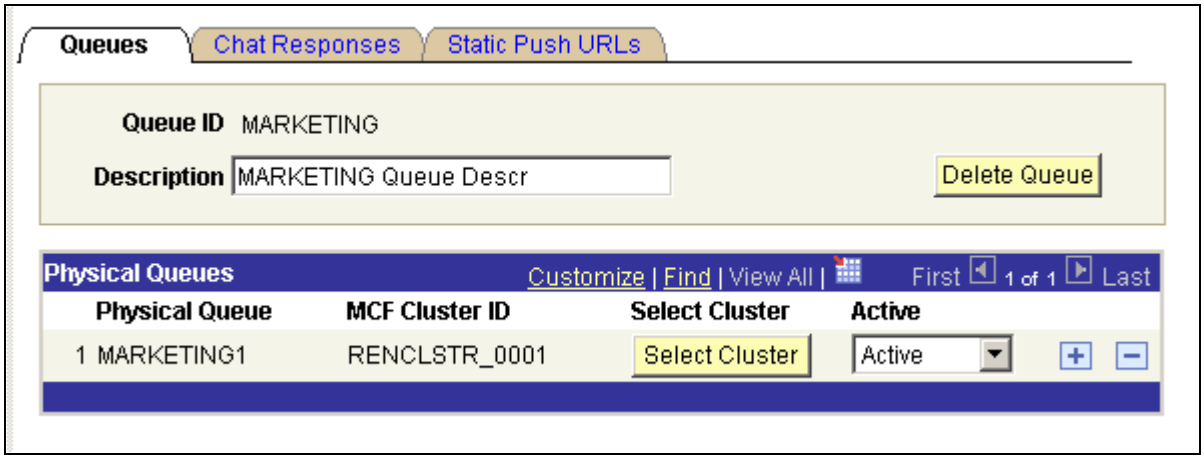
Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Chat Responses | MCFSYSMSG_PG | PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Queue, Chat Responses | Define chat messages that every agent can use. |
| Static Push URLs | MCFQUEUEURL_PG | PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Queue, Static Push URLs | Define URLs that each agent can push to a client. |

## Defining Queues

Access the Queues page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Queue, Queues




The Queues page showing the Queue ID and having the following editable fields: Description, Physical Queue, MCF Cluster ID, Select cluster and Active

**Queue ID** Queue IDs must be alphanumeric, cannot end in a numeral, but can include underscore characters.

**Delete Queue** Click to remove this queue.

Deleting a logical queue means that no work or agents can be assigned to the queue, and the queue is removed from all agents' available queues.

You can delete a logical queue only if all of its constituent physical queues are inactive and have no tasks. Verify that no application code assigns tasks to a queue before deleting the queue. All agents that are assigned to the child physical queue will receive a message notifying them to sign out of their MultiChannel Consoles when the logical queue is deleted.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Physical Queue** Identifies that part of a logical queue that is serviced by the selected MCF cluster.

Physical queue IDs always end in a number, for example, SALES2. The physical queue identifier automatically increments by one for each physical queue that is added.

Physical queue IDs are automatically generated. The maximum number of physical queues is nine.

**MCF Cluster ID** Identifies the MCF cluster that services this physical queue. Click Select Cluster to select from configured MCF clusters.

Each MCF cluster can service only one physical queue per logical queue. For example, an MCF cluster could service physical queue SALES1 or SALES2, but not both.

An MCF cluster can service multiple physical queues belonging to different logical queues. For example, an MCF cluster could service physical queues SALES1, MARKETING2, and COBOL1.

**Select Cluster** Click to select a cluster from a list of available MCF clusters.

The selected MCF cluster services this physical queue. The primary queue server in this cluster manages tasks and agents that are assigned to this physical queue.

**Active** Select *Active* or *Inactive* from the drop-down list box.

The queue server does not send new tasks to an inactive physical queue. Agents and existing tasks remain on an inactive physical queue. The physical queue must be active to receive new queued tasks.

Only inactive physical queues can be deleted from a logical queue. Inactivate a physical queue and complete or transfer all assigned tasks before deleting the physical queue.

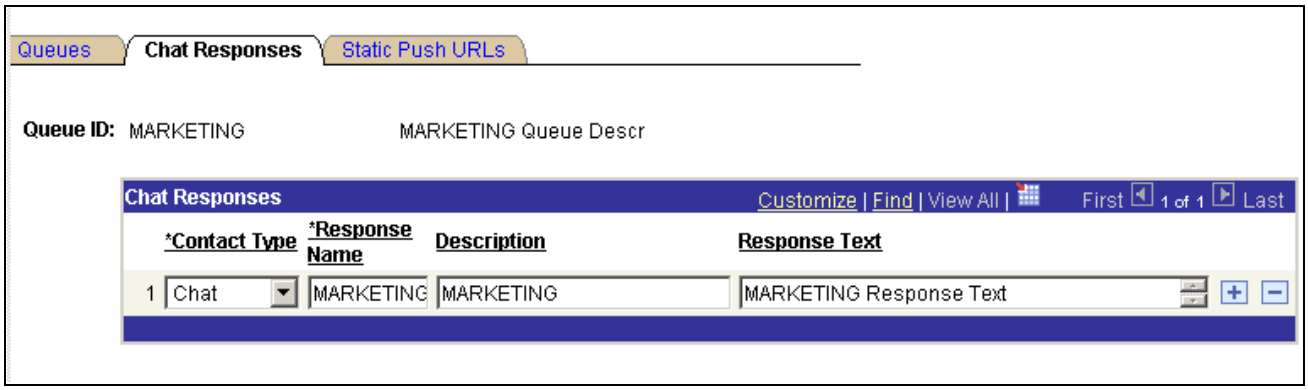
Active and inactive status support *follow-the-sun* practices. For example, an organization could support SALES1 in the London office, and SALES2 in the San Francisco office when the London office is closed by activating the appropriate queues.

## Defining Chat Responses

Access the Chat Responses page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Queue, Chat Responses

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Chat Responses page displaying the queue ID and having the following editable fields: Contact Type, Response Name, Description, Response Text

Chat responses that are specified on this page are available to all agents who are signed in to this queue.

**Contact Type** Select one of the following contact types:

- *Chat*
- *Email* (not currently supported)
- *Generic* (not currently supported)

**Response Name** The response name appears in the agent's Template Messages drop-down list box for all agents who belong to a physical queue on this logical queue

All chat responses are downloaded when the agent launches the agent chat console by accepting a customer chat. Template messages are not available in collaborative chat.

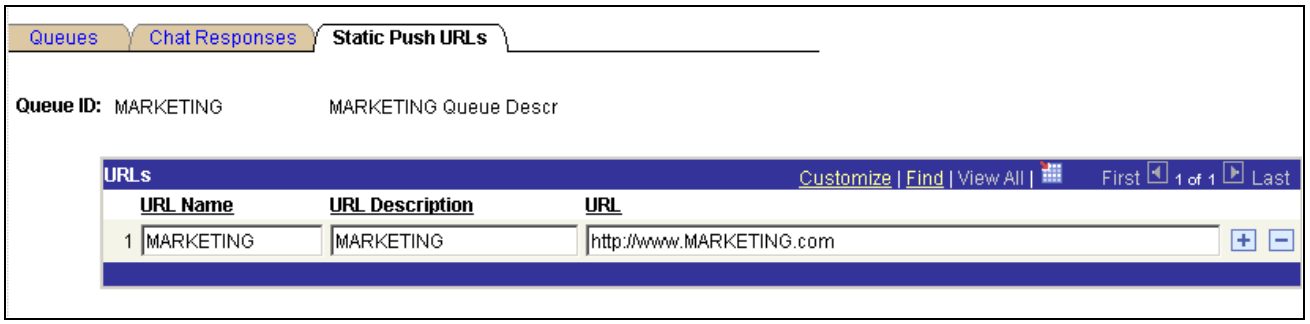
**Response Text** The specified message appears in the client chat window when selected by the agent.

## Defining Static Push URLs

Access the Static Push URLs page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Queue, Static Push URLs


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Static Push URLs page displaying the queue ID and having the following editable fields: URL Name, URL Description, and URL

Static URLs enable the agent to push a predefined URL to the customer, which appears in a pop-up window for the customer. The agent can edit these URLs in the chat window after selecting them.

**URL Name** Enter a name to identify the URL.

The URL name appears in the agent's Static URL drop-down list box for all agents who belong to this logical queue

**URL Description** Enter a description of the URL.

This description appears only on this page to further describe this URL or, for example, its reason for inclusion.

**URL** Enter the queue push URL.

The URL must include the opening http:// and any required parameters.

All static URLs that are defined for the queue are downloaded when the agent launches the agent chat console by accepting a customer chat. Static URLs are not available in collaborative chat.

If you send a static push URL that is a PeopleSoft Pure Internet Architecture URL, ensure that the recipient has permissions to access that portal, node, or page.

# Configuring Tasks

To configure tasks, use the MCF Task (MCF_TASKCFG_CMP) component.

This section discusses how to configure tasks.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

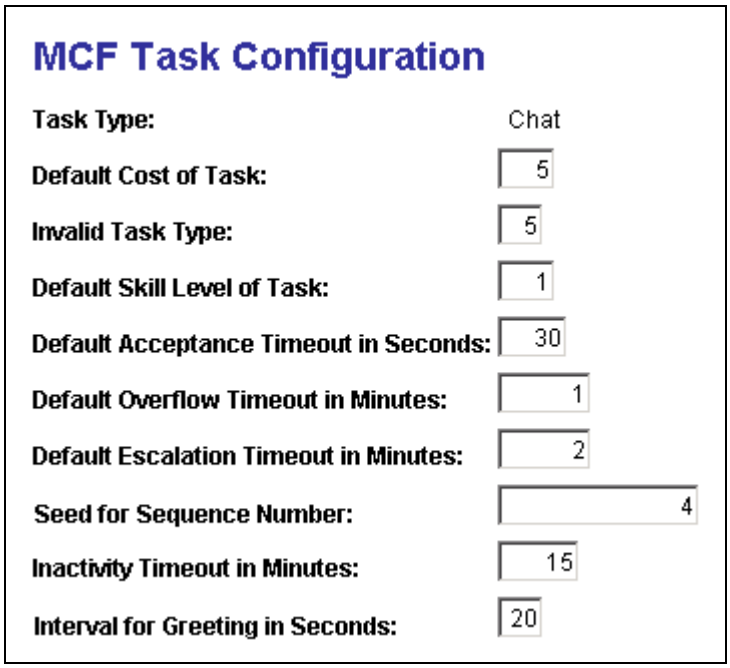
## Page Used to Configure Tasks

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| MCF Task Configuration | MCF_TASKCFG_PG | Use one of the following navigation paths:<br>• PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Tasks<br>• PeopleTools, Multichannel Framework, Third-Party Configuration, Tasks | Configure tasks such as chat, email, voice, and generic alerts using the queue server or third-party routing system. |

## Configuring Tasks

Access the MCF Task Configuration page using either of the following navigation paths whichever is appropriate to you:

PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Tasks (if you are using the CTI server) or PeopleTools,Multichannel Framework, Third-Party Configuration, Tasks (if you are using a third-party routing server).




The MCF Task Configuration page showing the task type and having the following editable fields: Default Cost of Task, Invalid Task Type, Default Skill Level of Task, Default Acceptance Timeout in Seconds, Default OverFlow Timeout in Minutes, Default Escalation Timeout in Minutes, Seed for Sequence Number, Inactivity Timeout in Minutes, and Interval for Greeting in Seconds


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Define values for different types of tasks that the queue server uses to assign tasks to appropriate agents and to manage tasks that are not accepted or closed within configurable time limits.

After you define or change values for a task, you must use the Refresh Task Properties button on the Cluster Notify page to propagate the changes to clusters. If you are working with MCF tasks, use the Cluster Notify page (MCF_CL_NOTIFY_PG) by navigating to PeopleTools, Multichannel Framework, Universal Queue, Configuration, Notify Cluster. If you are configuring third-party tasks, use the third-party Cluster Notify page (MCFTP_CL_NOTIFY_PG) by navigating to PeopleTools, Multichannel Framework, Third-Party Configuration, Cluster Notify.

**Default Cost of Task** Enter the cost of the task.

Cost is a measure of the workload that each task places on an agent. The cost of a task is an estimate of the task's expected complexity and of the time that is required to resolve the task. The minimum value is *0*, and no maximum value exists.

The costs of tasks that are assigned to an agent are added up and evaluated against the maximum workload for each agent to determine whether the agent can receive additional tasks. For example, if an agent has a maximum workload of 100, and the default cost of a chat is 20, the agent can manage five concurrent chat sessions, assuming that the default cost is not overridden in the InitChat() built-in function call and that no other task types have been assigned.

---

**Note.** Although priority has no effect on voice tasks (which are not queued), voice task cost is included in calculating an agent's workload.

---

Default costs are:

- Chat: *5*
- Email: *2*
- Generic: *1*
- Voice: *10*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Default Priority of Task**

Enter the priority of this task. A higher value means a higher priority. Tasks are ordered on a physical queue based on their assigned priority.

The minimum value is *0*, and no maximum value exists.

A queue server gives precedence to a task of higher-priority value over a task of lower-priority value when looking for an agent to assign the task to. This means that the queue server always assigns a task of priority 100 to a qualified available agent before it looks for an agent for a task of priority 10. If two tasks have the same priority, they are assigned in the order of their enqueue time.

The value that is specified here can be overridden in the EnQueue() or InitChat() built-in function call.

---

**Note.** Priority has no effect on voice tasks, which are not queued; however, voice task cost is included in calculating an agent's workload.

---

Default priorities are:

- Chat: *5*
- Email: *2*
- Generic: *1*
- Voice: *10*

**Default Skill Level of Task**

Enter the minimum agent skill that is required to handle this task.

The queue server assigns this task type to an available agent with the lowest skill level on that queue that is greater than or equal to the skill level that is required by the task.

The minimum value is *0*, and no maximum value exists.

The value that is specified here can be overridden in the EnQueue() or InitChat() built-in function call.

Default skill levels are:

- Chat: *1*
- Email: *1*
- Generic:*1*
- Voice : *2*

---

**Note.** Only the third-party routing server supports voice channel. The queue server does not route voice tasks.

---


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Default Acceptance Timeout in Seconds**

Specify the period of time that an agent has to accept an assigned task (to click the flashing icon on the MultiChannel Console). If the task is not accepted within this time, the task is reenqueued for assignment to another agent.

The queue server uses an algorithm to minimize reassignment of tasks that previously timed out to the same agent.

The value that is specified here can be overridden in the EnQueue() or InitChat() built-in function call.

Default acceptance timeouts are:

- Chat: *30*
- Email: *30*
- Generic: *30*
- Voice : *10*

**Note.** Only the third-party routing server supports voice channel. The queue server does not route voice tasks.

**Default Overflow Timeout in Minutes**

Specify the overflow time-out.

The overflow time-out is the time period that a queue server has to find an agent who accepts a task (click the flashing icon on the MultiChannel console). If the task is not accepted within this time, the task is removed from the queue and placed in the overflow table. This table can be managed from the Overflow Administration page.

The value that is specified here can be overridden in the EnQueue() or InitChat() built-in function call.

Default overflow time-outs are:

- Chat: *2*
- Email: *120*
- Generic: *120*
- Voice: *1*

**Note.** Only the third-party routing server supports voice channel. The queue server uses CTI to route voice tasks.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Default Escalation Timeout in Minutes**

Specify the default escalation time-out.

The escalation time-out is the time period within which a task must be closed. If the task is not closed within this time, the task is removed from the queue and from the agent's accepted task list (that is, the task is unassigned) and the task is placed in the escalation table. This table can be managed from the Escalation Administration page.

Escalation time-out is valid on a chat session only after it is accepted by an agent, and has no effect on voice tasks (CTI).

The value that is specified here can be overridden in the EnQueue() or InitChat() built-in function call.

Default escalation time-outs are:

- Chat: *10*
- Email: *480*
- Generic: *480*
- Voice: *2*

**Note.** Only the third-party routing server supports voice channel. The queue server uses CTI to route voice tasks.

**Seed for Sequence Number**

Displays the current cumulative count of tasks of this type that are enqueued. Do not modify this value until it has reached its upper limit of 2,147,483,647.

This value does not apply for the voice task type.

**Inactivity Timeout in Minutes**

Specify the inactivity time-out

Inactivity time-out applies to chat only. The inactivity timeout is the time period within which an agent or customer must participate in a chat. If the chat session is dormant for more than this time, the chat is terminated.

The default value is *15* minutes.

**Interval for Greeting in Seconds**

Specify the interval, in seconds, over which the initial greeting appears in a customer chat window while the system searches for an available agent.

**Note.** All task parameters are delivered with sample values. Determine a range for these values that is appropriate to your business requirements. For example, task cost could vary over a range of 1 to 100 instead of 1 to 10.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

***See Also***

Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116

Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Notifying PeopleSoft MCF Clusters of Changed Parameters for a Third Party, page 215

Chapter 8, "Configuring PeopleSoft MCF Agents," Creating Agents, page 120

# Viewing the Cluster Summary

To view the Cluster Summary page, use the Cluster Summary (MCF_RSERV_CFG_CMP) component.

This section discusses how to view the Cluster Summary page.

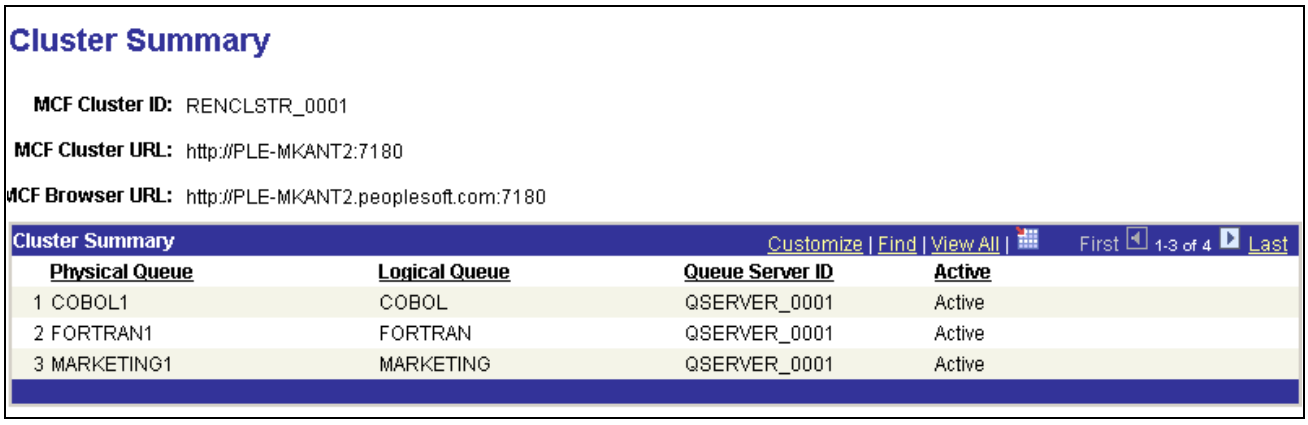
## Page Used to View the Cluster Summary

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Cluster Summary | MCF_RENSRV_PG | PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Cluster Summary | View summary information about MCF clusters. |

## Viewing the Cluster Summary

Access the Cluster Summary page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Cluster Summary




Cluster Summary page showing the MCF Cluster ID, MCF Cluster URL, and MCF Browser URL. It shows the following cluster summary details: Physical column, Logical Column, Queue Server ID, and Active.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

The Cluster Summary page displays the associated MCF cluster URLs and queue details for the selected MCF cluster. The information cannot be changed from this page.

# Tuning Cluster Parameters

To tune cluster parameters, use the Cluster Tuning (MCF_SYSTEM_NV_CMP) component.

This section discusses how to tune cluster parameters.

## Page Used to Tune Clusters

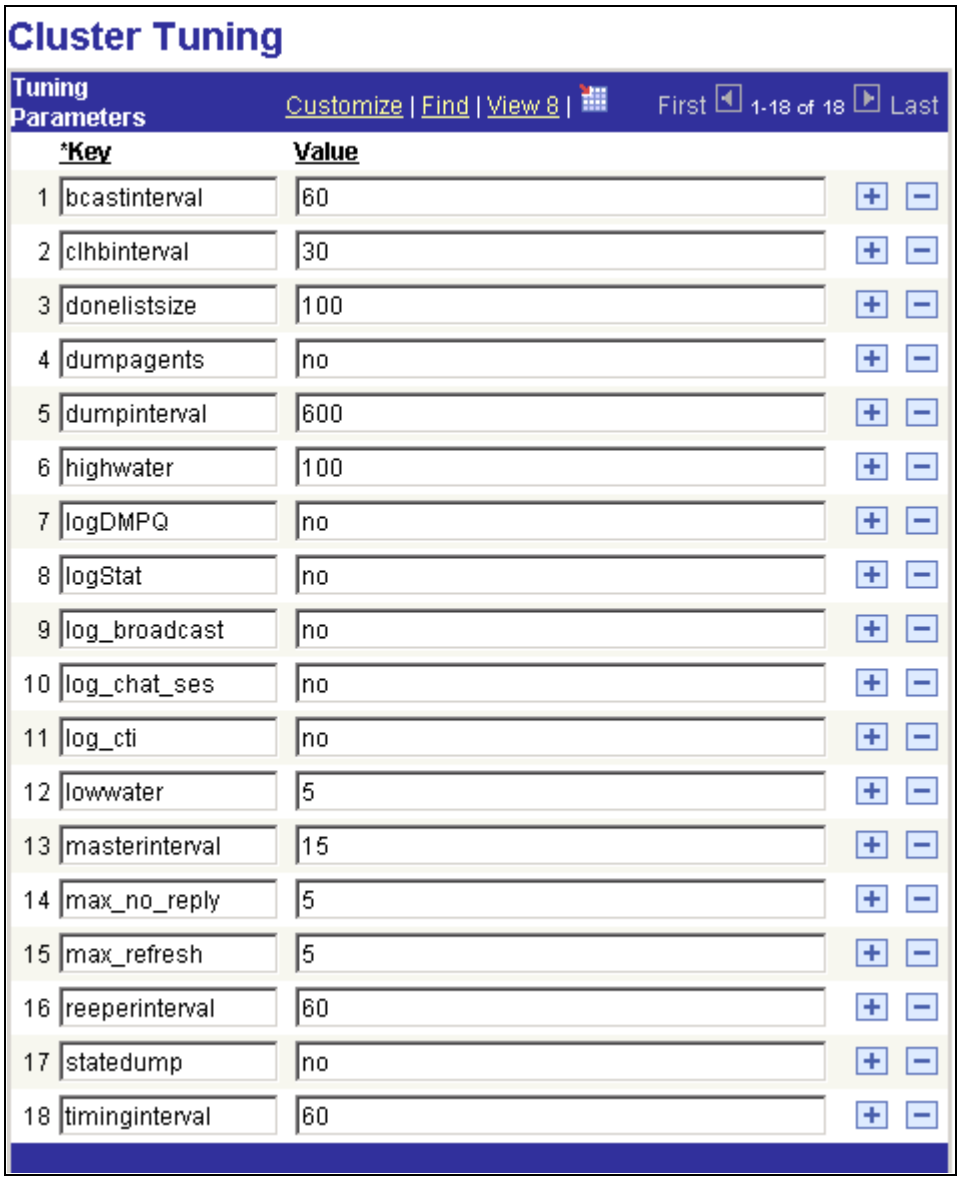
| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Cluster Tuning | MCF_SYSTEM_KV_PG | PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Cluster Tuning | Modify MCF cluster parameters to tune performance. |

## Tuning Cluster Parameters

Access the Cluster Tuning page using the following navigation path:

PeopleTools ,MultiChannel Framework, Universal Queue, Configuration, Cluster Tuning

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Cluster Tuning page showing the Key and Value parameters

Use the Cluster Tuning page to set MCF cluster parameters to optimize performance or enable logging.

If you make changes to a cluster parameter, you must use the Notify Cluster page to propagate the changes.

See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116.

The following table lists the cluster tuning parameters you can modify and describes the default values and usage of each.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Key* | *Default value* | *Usage* |
|---|---|---|
| bcastinterval | *60* | The interval, in seconds, after which the number of unassigned tasks per physical queue and the number of agents that are logged into each physical queue are broadcast to the MultiChannel Consoles for display next to the queue names.<br><br>A smaller value provides more accurate queue statistics, but increases the load on the queue server and REN server. A larger value decreases queue server and REN server load, but also decreases statistical accuracy.<br><br>The bcastinterval value also determines how frequently onStat2 event statistics are calculated. A smaller value provides updated statistics more frequently.<br><br>This is a timing parameter. If you change the value of this parameter, you must use the Refresh Threshold/Timing Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br><br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116; Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the Monitor Queues Page and Sample Monitor - Queue Statistics Page, page 197 and Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the Monitor Agents Page and Sample Monitor - Agent States Page, page 195. |
| clhbinterval | *30* | The interval, in seconds, in which the queue server expects to receive a heartbeat from a connected JSMCAPI client. If the queue server receives no heartbeat during this interval, the queue server stops the client session. For a queue server, to avoid a session time-out for the client when CPU usage of the machine on which the client is running is high, increase the heartbeat interval of the client.<br><br>**Note.** For third-party server, increase the heartbeat interval on the third-party server side to avoid a session time-out.<br><br>This is a timing parameter. If you change the value of this parameter, you must use the Refresh Threshold/Timing Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br><br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116 and Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using and Demonstrating JSMCAPI, page 178. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Key* | *Default value* | *Usage* |
|---|---|---|
| donelistsize | *100* | The number of completed tasks that are stored in the list that is used to calculate average task duration.<br>Configure the donelistsize value depending on the task volume that is encountered and the interval over which you want to monitor tasks.<br>This is a timing parameter. If you change the value of this parameter you must use the Refresh Threshold/Timing Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116 and Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using and Demonstrating JSMCAPI, page 178. |
| dumpagents | *No* | Enter *Yes* if the status of agent activity should be written to the database during the periodic state dumps.<br>Logging agent status increases queue server load, but provides information about agent performance.<br>This is a task parameter. If you change the value of this parameter, you must use the Refresh Task Properties button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116 and Chapter 9, "Administering Queues, Logs, and Tasks," Viewing the Agent State Summary, page 139. |
| dumpinterval | *600* | The interval, in seconds, after which the queue state is written to the database.<br>A smaller value increases load on the queue server, but provides more frequent statistics. A larger value decreases load on the queue server, but provides less frequent statistics.<br>A value of less than one minute will significantly reduce queue server performance.<br>This is a timing parameter. If you change the value of this parameter, you must use the Refresh Threshold/Timing Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116; Chapter 9, "Administering Queues, Logs, and Tasks," Viewing the Queue Server State, page 137 and Chapter 9, "Administering Queues, Logs, and Tasks," Viewing the Queue State Summary, page 138. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Key* | *Default value* | *Usage* |
|---|---|---|
| highwater | *100* | The maximum number of persistent tasks that are retrieved from the database and cached in memory in the queue server.<br><br>The highwater and lowwater mark values determine how often, and how many, persistent tasks should be read into memory.<br><br>A higher value causes the queue server to retrieve more persistent tasks at one time, which results in less frequent access to the database, but also more tasks for the queue server to manage, slowing performance. A lower value speeds performance, but requires more frequent access to the database.<br><br>If a large number of enqueued tasks cannot be routed by the queue server (for example, a call center handling a specific physical queue is offline), increase the highwater mark so that tasks that can be routed can fit into memory cache.<br><br>This is a threshold parameter. If you change the value of this parameter, you must use the Refresh Threshold/Timing Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br><br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116. |
| logDMPQ | *No* | Enter *Yes* if you want PSMCFLOG to log REN server event notifications resulting from bcastinterval broadcasts.<br><br>Only the event is logged, not its contents.<br><br>This is a logging parameter. If you change the value of this parameter, you must use the Refresh Logging Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br><br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116 and Chapter 9, "Administering Queues, Logs, and Tasks," Viewing Event Logs, page 144. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Key* | *Default value* | *Usage* |
|---|---|---|
| logStat | *No* | Enter *Yes* to log the statistics that are returned by the queue server for the onStat1 user and group events to the database.<br>This is a logging parameter. If you change the value of this parameter, you must use the Refresh Logging Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116 and Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using and Demonstrating JSMCAPI, page 178. |
| log_broadcast | *No* | Enter *Yes* to activate logging of the broadcast messages that are sent.<br>This is a logging parameter. If you change the value of this parameter, you must use the Refresh Logging Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116 and Chapter 9, "Administering Queues, Logs, and Tasks," Viewing Broadcast Logs, page 141. |
| log_chat_ses | *No* | Enter *Yes* to activate logging of the contents of chat sessions.<br>This is a logging parameter. If you change the value of this parameter, you must use the Refresh Logging Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116 and Chapter 9, "Administering Queues, Logs, and Tasks," Viewing Chat Logs, page 142. |
| log_cti | *No* | Select *Yes* to activate logging of CTI events.<br>This is a logging parameter. If you change the value of this parameter, you must use the Refresh Logging Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116 and Chapter 3, "Configuring PeopleSoft Computer Telephony Integration," Logging CTI Events, page 47. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Key* | *Default value* | *Usage* |
|---|---|---|
| lowwater | *5* | The minimum number of persistent tasks that are cached in memory in the queue server. When the lowwater value is reached, the queue server retrieves another batch of persistent tasks, up to the highwater value.<br>The highwater and lowwater mark values determine when, and how many, persistent tasks should be read into memory.<br>A higher value requires the queue server to access the database more frequently. A lower value can cause the queue server to run out of persistent tasks before refreshing its queue.<br>The lowwater value should be greater than or equal to the maximum number of agents that are logged onto any physical queue at one time.<br>This is a threshold parameter. If you change the value of this parameter, you must use the Refresh Threshold/Timing Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116. |
| masterinterval | *15* | The interval, in seconds, after which a cluster master updates its timestamp in its cluster tables. Slave clusters check the timestamp to determine whether the master cluster is still running.<br>A lower value enables rapid discovery of a failed master server, but increases queue server overhead. A higher value reduces queue server overhead, but delays discovery of a failed master server.<br>If only one queue server is configured for an MCF cluster, this value can be large.<br>The masterinterval value also acts as a heartbeat interval for the master queue server connection to user consoles.<br>This is a timing parameter. If you change the value of this parameter, you must use the Refresh Threshold/Timing Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116. |

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Key* | *Default value* | *Usage* |
|---|---|---|
| max_no_reply | *5* | Sets the maximum number of consecutive agent timeouts before the queue server automatically signs out the agent and sets the agent's console status as Assumed Unavailable.<br>This is a timing parameter. If you change the value of this parameter, you must use the Refresh Threshold/Timing Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116. |
| max_refresh | *5* | Sets the maximum number of consecutive times that results are discarded when task queue is refreshed from the database if an intervening notification of new persistent tasks exists.<br>This is a timing parameter. If you change the value of this parameter, you must use the Refresh Threshold/Timing Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116. |
| reeperinterval | *60* | The interval, in seconds, after which deleted tasks are cleared from memory in the queue server.<br>A lower value increases queue server load but clears memory more frequently. A higher value decreases queue server load but clears memory less frequently.<br>This is a timing parameter. If you change the value of this parameter, you must use the Refresh Threshold/Timing Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Key | Default value | Usage |
|---|---|---|
| statdump | *No* | Specify *Yes* to write queue server state to the database during the periodic state dumps. The state dump interval is set by the dumpinterval parameter.<br>This is a task parameter. If you change the value of this parameter, you must use the Refresh Task Properties button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116 and Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using and Demonstrating JSMCAPI, page 178. |
| timinginterval | *60* | The interval, in seconds, after which the database is checked for expired or overflowed persistent tasks. This parameter does not affect real-time tasks.<br>A lower value increases queue server load but detects timed-out tasks more quickly. A higher value decreases queue server load but detects timed-out tasks less frequently.<br>This is a timing parameter. If you change the value of this parameter, you must use the Refresh Threshold/Timing Parameters button on the Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Notifying Clusters of Changed Parameters, page 116. |

# Notifying Clusters of Changed Parameters

To notify clusters of changed parameters, use the Cluster Notify (MCF_AD_NOTIFY_CMP) component.

This section discusses how to notify clusters of changed parameters.

## Page Used to Notify Clusters of Changed Parameters

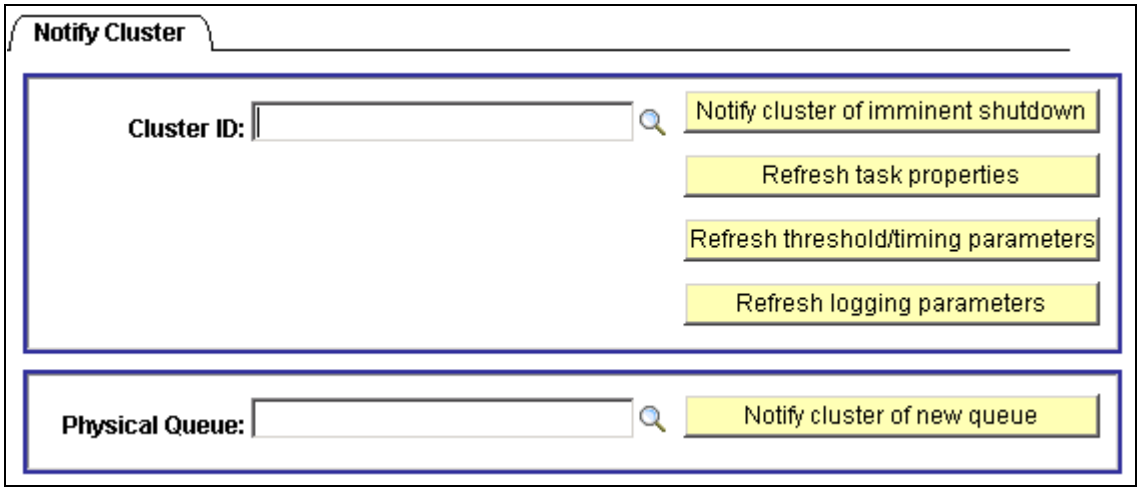
| Page Name | Definition Name | Navigation | Usage |
|---|---|---|---|
| Notify Cluster | MCF_CL_NOTIFY_PG | PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Cluster Notify | Notify a cluster of changes to parameters or configuration, or of shutdown. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Notifying Clusters of Changed Parameters

Access the Notify Cluster page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Cluster Notify




The Notify Cluster page having the Cluster ID and Physical editable fields

Use the Notify Cluster page to notify an MCF cluster of certain changes to its parameters or constituent queues, or that its application servers are being shut down.

For example, after changing MCF cluster parameters on the Cluster Tuning page, use the Notify Cluster page to refresh the tuning parameters.

| | |
|---|---|
| **Notify cluster of imminent shutdown** | Click to send a message to all agents who are signed in to the selected MCF cluster that they have been signed out.<br>Send this notification if the cluster's application servers are being shut down. |
| **Refresh task properties** | Click to load task properties that have been changed on the Tasks page for the selected MCF cluster. |
| **Refresh threshold/timing parameters** | Click to reload threshold and timing parameters that have changed on the Cluster Tuning page for the selected MCF cluster. |
| **Refresh logging parameters** | Click to reload logging parameters that have changed on the Cluster Tuning page for the selected MCF cluster. |
| **Notify cluster of new queue** | Click to notify the selected MCF cluster that the selected physical queue has been added. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 8

# Configuring PeopleSoft MCF Agents

This chapter discusses how to:

- Define agents.
- Define optional agent characteristics.

## Defining Agents

To define agents, use the MCF Agent (MCF_AGENT_CMP) component.

This section discusses how to:

- Create agents.
- Specify languages that an agent supports.
- Personalize an agent's presence.

The previous three agent definition pages form the basis of agent configuration. Other parameters are optional.

### Pages Used to Define Agents

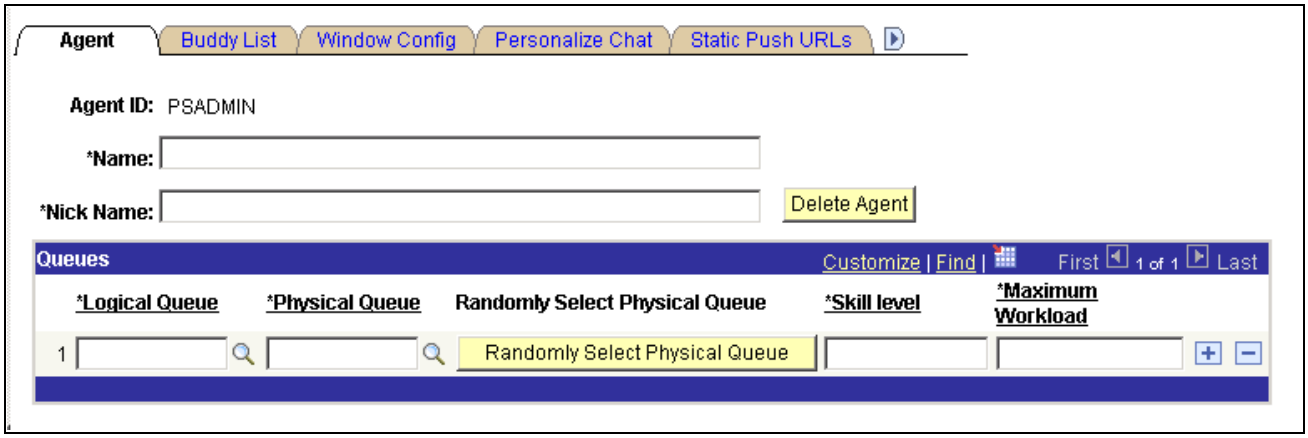
| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Agent | MCF_AGENT_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Agent | Create agents.<br>Specify name, skill level, maximum workload, and queues for each agent. |
| Languages | MCF_AGENTLANG_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Languages | Specify the languages that an agent is qualified to use. Specify at least one language. |
| Personalize Presence | MCFAGENTPRES_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Personalize Presence | Specify an agent's presence options. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Creating Agents

Access the Agent page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Agent




The Agent page displaying the Agent ID and having the following editable fields: Name, Nick Name, Logical Queue, Physical Queue, Randomly Select Physical Queue, Skill Level, and Maximum Workload.

**Name**

Enter the full name, in (lastname,firstname) format, of this agent.

The agent name appears in other agents' buddy lists.

**Note.** There is no space in between lastname, firstname.

**Nick Name**

Enter a short name for this agent.

The agent nickname identifies this agent in chat sessions and logs.

**Delete Agent**

An agent cannot be deleted if the agent still has accepted tasks on any queues to which the agent belongs. Before deleting an agent, ensure that the agent is logged off from all queues to which the agent is assigned.

**Logical Queue ID**

Enter the ID of a logical queue to which this agent is assigned.

Each agent can be assigned to more than one logical queue. An agent can log on to only one queue at a time from the MultiChannel Console.

**Note.** Do not overwrite the logical queue except when first creating an agent, as the agent's tasks may lose their assignments.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Physical Queue ID**

Agents are randomly assigned to a physical queue when the logical queue is associated with the agent. An agent who services a logical queue logs on to a physical queue that is managed by a specific MCF cluster.

While an agent can service multiple logical queues, the agent can belong to only one physical queue per logical queue.

**Note.** Do not overwrite the physical queue except when first creating an agent, as this may orphan tasks. Use the Physical Queues Move Agent page.

**Randomly Select Physical Queue**

Click to assign another physical queue (within this logical queue) randomly. This selection will help to spread multiple agents evenly over available physical queues.

**Skill Level**

Select the skill level of this agent for the tasks that are assigned for this queue. This field is required.

The agent is assigned only tasks requiring a skill level that is less than or equal to the skill level specified here. If more than one qualified agent is available to accept the task, the queue server gives preference to the agent with the lowest skill level.

Each agent can have a different skill level for each queue to which the agent is assigned.

**Maximum Workload**

Select the maximum load that this agent can be assigned before tasks are held or assigned to other agents. This field is required.

The cost of each accepted task is added to the agent's current workload. A task is not assigned to an agent if its cost pushes the agent's current workload over the maximum.

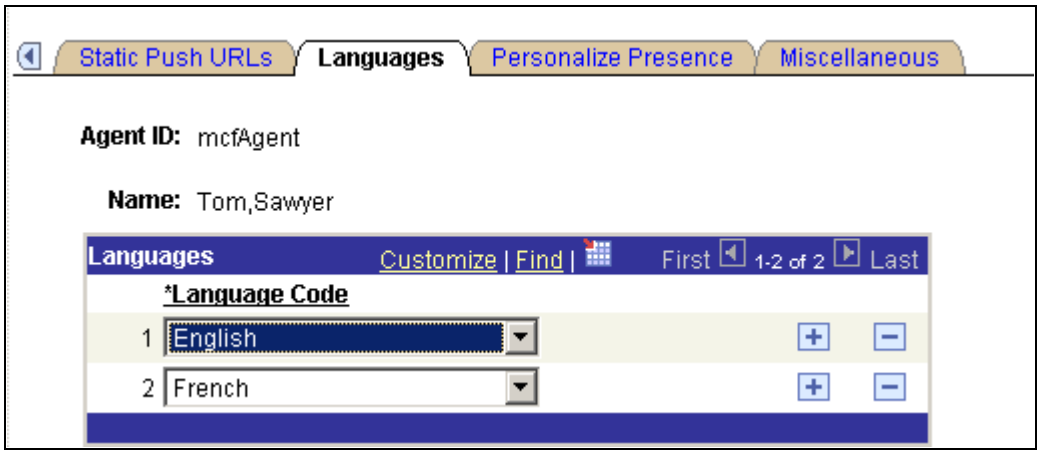
**Note.** Do not delete a queue from an agent's list unless that agent has no open accepted tasks in that queue.

## Specifying Languages That an Agent Supports

Access the Languages page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Languages

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Languages page displaying the Agent ID and Name and having the Language Code editable field.

Specify the languages that each agent is qualified to support.

The agent is assigned only tasks that have been enqueued with a language code in the agent's language list. For the EnQueue() built-in function, the language code is specified as a parameter. For the InitChat() built-in function, the language code is determined by the user profile of the initiator.

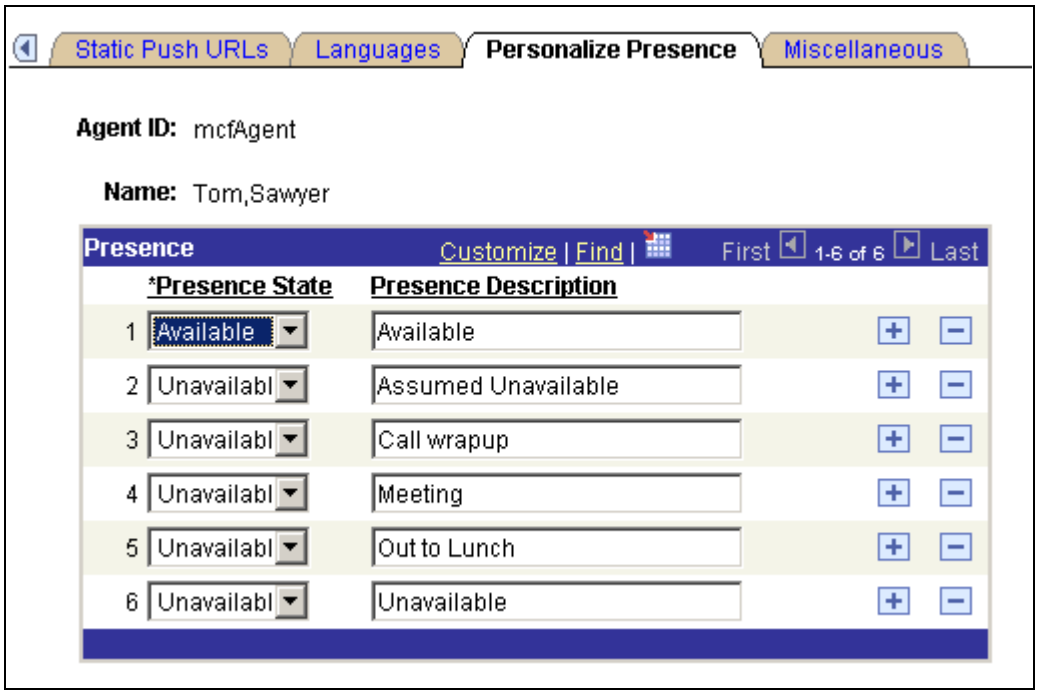
If you do not specify a language code for a new agent, the default value is *English.*

## Personalizing an Agent's Presence

Access the Personalize Presence page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Personalize Presence


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Personalize Presence page displaying the Agent ID and Name and having the Presence State and Presence Description editable fields.

Each agent can configure the presence description that is displayed when the agent is available or unavailable. The queue server understands only the presence state, available or unavailable, but you can specify more specific presence descriptions when displaying or logging an agent's presence. For example, Lunch, Meeting, or Indisposed are unavailable states that can be used for tracking agent time and activity.

If you do not specify presence descriptions, default values are used.

**Presence State** Select *Available* or *Unavailable.*

**Presence Description** Enter a description for each agent state.

The description appears in logs of agent activity and when agent presence is displayed.

**Note.** The *Available* state has only one description. For the *Unavailable* state, you can enter any presence description, such as tea break, coffee break, lunch, and so on.

# Defining Optional Agent Characteristics

To define optional agent characteristics, use the MCF Agent (MCF_AGENT_CMP) component.

This section discusses how to:

- Set up buddy lists.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Configure windows.
- Personalize chat.
- Specify agent-specific URLs.
- Specify miscellaneous parameters.

The agent configuration pages are considered optional because most do not have default values and can remain unconfigured without affecting an agent's ability to log on to a queue and accept tasks.

## Pages Used to Define Optional Agent Characteristics

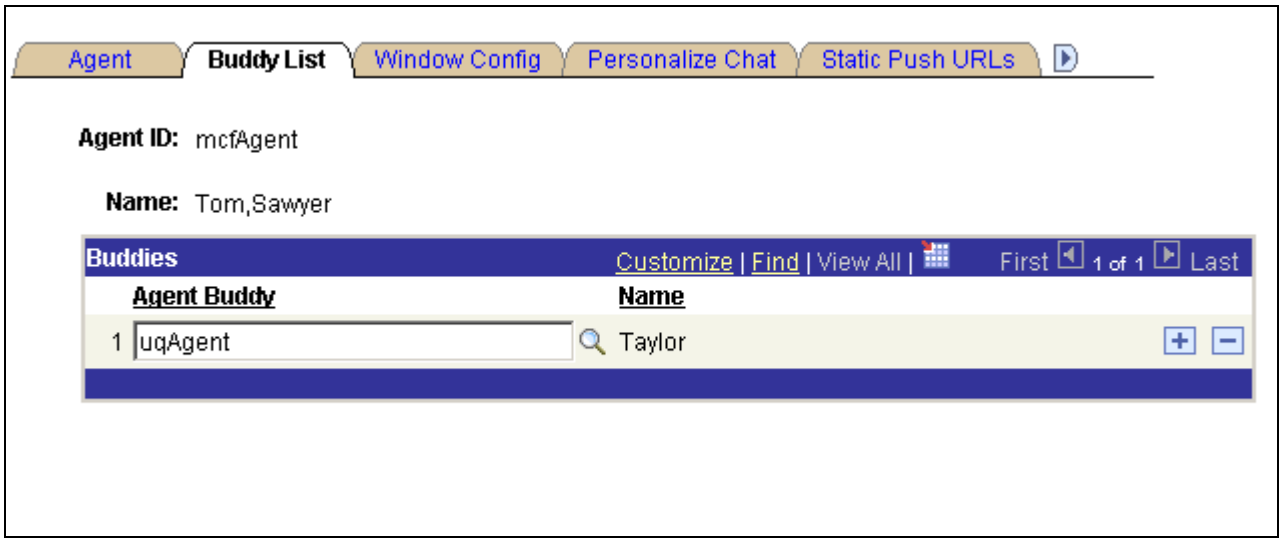
| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Buddy List | MCFAGENTBUDDY_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Buddy List | Specify a list of other agents. |
| Window Config (window configuration) | MCFAGENTCUST_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Window Config | Configure the specified agent's window. |
| Personalize Chat | MCFAGENTMSG_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Agent Messages | Specify agent-specific messages, in addition to available system messages. |
| Static Push URLs | MCFAGENTURL_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Agent Push URLs | Specify agent-specific URLs, in addition to system push URLs. |
| Miscellaneous | MCF_AGENTMISC | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Miscellaneous | Specify miscellaneous behaviors. |

## Setting Up Buddy Lists

Access the Buddy List page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Buddy List


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Buddy List page displaying the Agent ID and Name and having the Agent Buddy and Name editable fields.

The agent's buddy list facilitates collaborative chat and chat conferencing.

**Agent Buddy** Select another agent with whom this agent can have a chat session or can ask to conference into another chat.

Each agent buddy must be logged in a physical queue on the same cluster to be able to chat. If two agents must be able to chat but they do not share a cluster, use the Physical Queue Move Agent page to move the agents into physical queues on the same MCF cluster.

Agent buddies are listed with their login status in the buddy list on the multichannel console.

**Name** Displays the buddy agent's nickname.

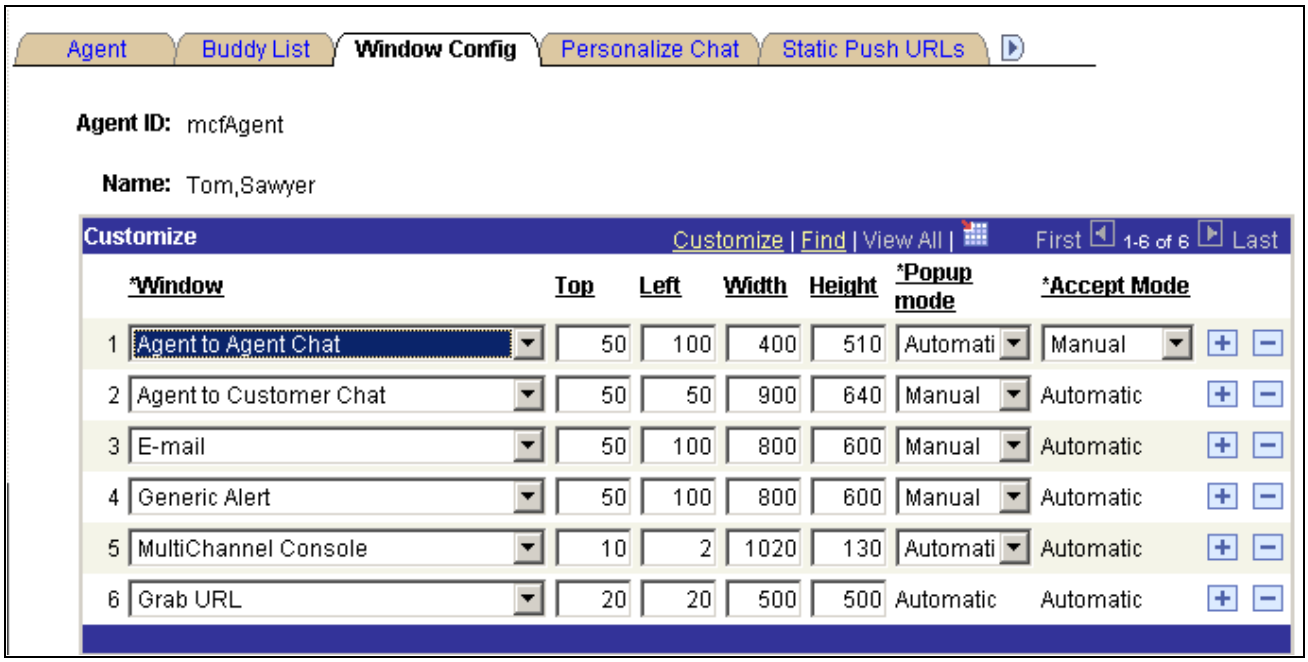
An agent's presence, as shown in the buddy list on the MultiChannel Console or on the Invite Agent list on the chat console, indicates the agent's availability for chat or conference.

## Configuring Windows

Access the Window Config page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent,Window Config

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Window Config page displaying the Agent ID and Name and having the following editable fields: Window, Top, Left, Width, Height, Popup Mode, and Accept Mode.

Set the initial agent window placement and size by specifying parameters on this page. An agent can resize and move the windows.

**Window** Select the window to which the specified configuration applies.

Select from:

- *Agent to Agent Chat*
- *Agent to Customer Chat*
- *E-mail*
- *Generic Alert*
- *Grab URL*
- *MultiChannel Console*

**Top** and **Left** Enter the distance in pixels from the top and left edges of the screen when the window first appears.

**Width** and **Height** Enter the width and height, in pixels, of the window when it first appears.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Popup Mode** Select from:

*Automatic:* The window appears automatically. For customer-initiated chat or tasks that are initiated from the EnQueue() built-in function, the task is automatically accepted as well. For agent-initiated chat, the agent can elect not to accept the task; in effect, the agent can preview the task. If this is the desired behavior, select *Manual* from the Accept Mode drop-down list box. If you want agent-to-agent tasks to function like customer-initiated tasks, select *Automatic* from the Accept Mode drop-down list box.

*Manual:* The window does not appear until the agent clicks the task on the agent MultiChannel Console. For customer-initiated chat or tasks that are initiated from the EnQueue() built-in function, clicking the task means that the task is automatically accepted as well. For agent-initiated chat, the behavior depends on the setting for the Accept Mode field.

**Accept Mode** Select from:

*Automatic:* Agent-to-agent chats are automatically accepted without requiring the agent to click the icon.

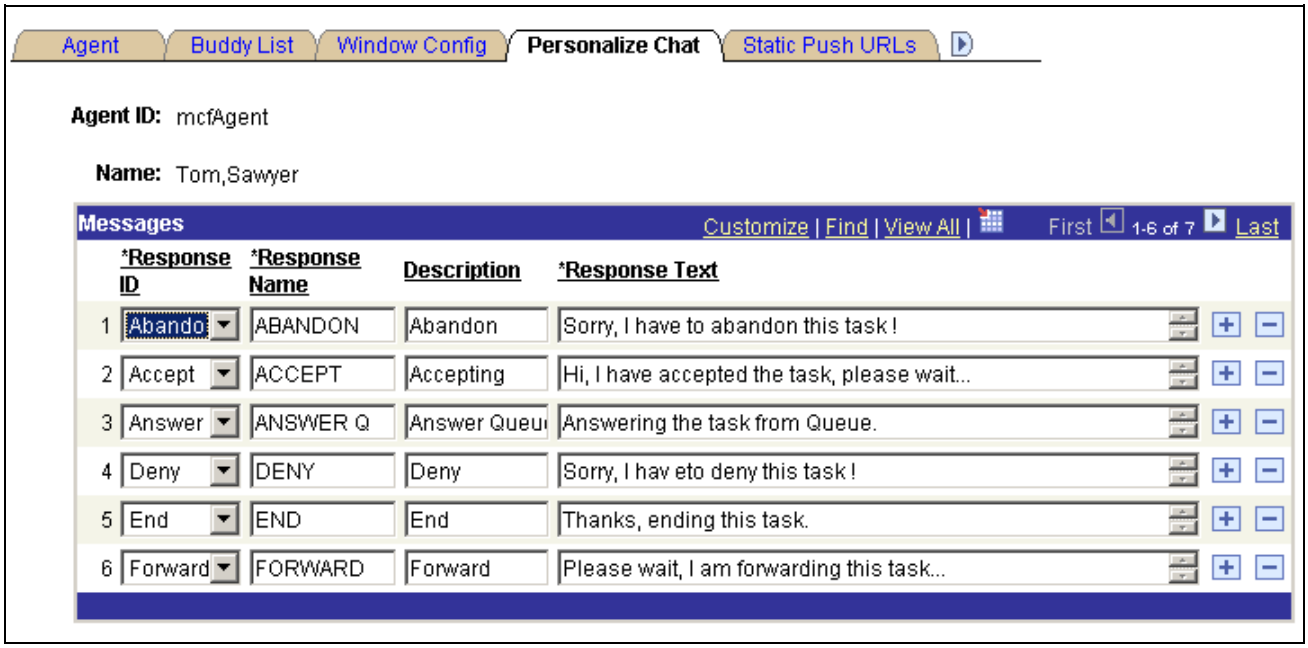
*Manual:* Agent-to-agent chats require the agent to click the icon.

Accept mode affects only collaborative chat.

## Personalizing Chat

Access the Personalize Chat page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Agent Messages




The Personalize Chat page displaying the Agent ID and Name and having the following editable fields: Response ID, Response Name, Description, and Response Text.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

An agent can create personalized responses in addition to the system responses that are defined for each queue.

**Response ID** Responses, except those that are identified by *Other*, are linked to specific events. These responses are always sent on these events from this agent. If an agent does not have a customized response for a specific event, the response is read from a default value that is set in the Message Catalog. The response text that is set here overrides the default text that is set in the Message Catalog.

Select from:

- *Abandon:* A chat is abandoned when a chat initiator closes the chat window before the chat is accepted by an agent. This message appears when the agent accepts the abandoned chat.
- *Accept:* This response is automatically sent to the chat initiator when an agent accepts a chat in response to a chat request that does not include a question.

  If the chat request includes a question, the agent's *Answer Question* text is sent in response instead of the *Accept* response.
- *Answer Question:* This response is automatically sent to the chat initiator when an agent accepts a chat in response to a chat request that includes a question.
- *Deny:* This response applies only to collaborative chat. If an agent elects not to accept a chat, this message is automatically sent to the chat initiator.
- *End:* If either party quits a chat after the chat is accepted, this message is displayed from the agent.
- *Forward:* If the agent forwards a chat session to another queue, this message is sent to the customer.
- *Other:* These messages are never automatically sent in a chat session. Their message names appear in the Template Messages drop-down list box on the agent chat page. These messages are appended to the template messages (chat responses) that are defined for the queue.

**Response Name** This name appears in the agent's template response drop-down list box.

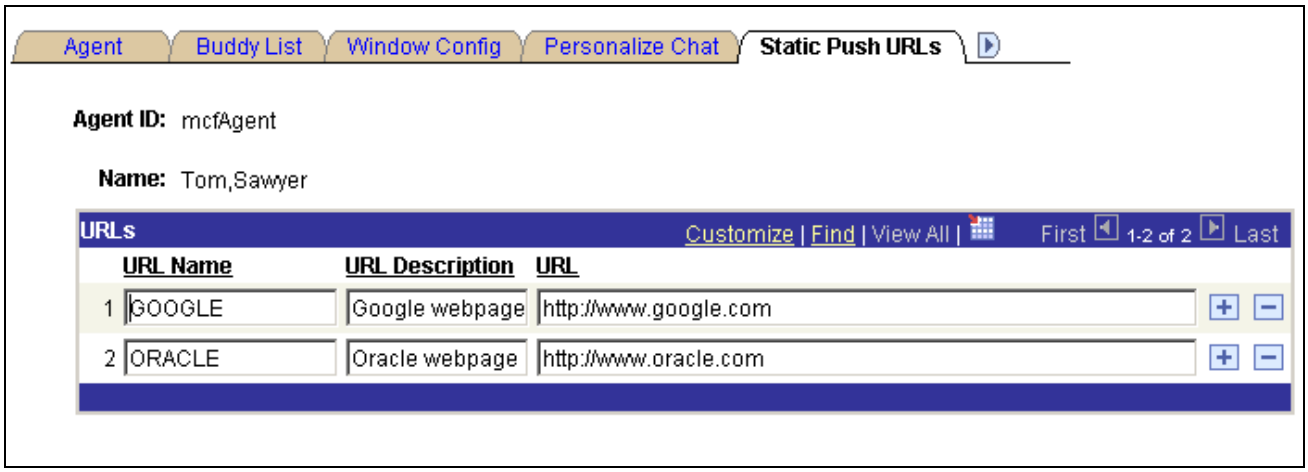
**Response Text** Enter the response text to appear in the chat window.

## Specifying Agent-Specific URLs

Access the Static Push URLs page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Agent Push URLs

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Static Push URLs page displaying the Agent ID and Name and having the following editable fields URL Name, URL Description, and URL.

This page defines URLs that this agent can send to a client browser. These URLs are in addition to the URLs that are defined in the queue configuration page.

**URL Name** The URL name appears in the agent's static URL drop-down list box.

**URL Description** This description appears only on this page, to further describe this URL or, for example, its reason for inclusion.

**URL** Enter the queue push URL.

The URL must include the opening http:// and any required parameters.

All static URLs that are defined for the agent are downloaded when the agent launches the agent chat console by accepting a customer chat. Static URLs are not available in collaborative chat.

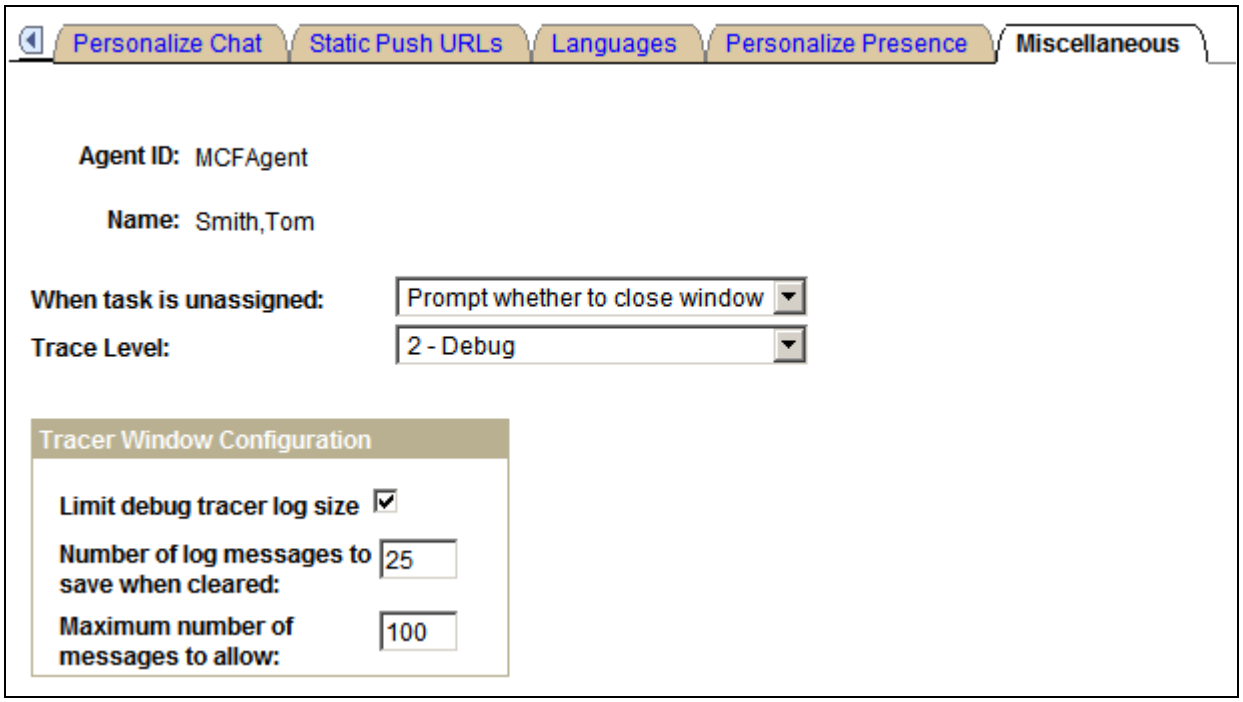
If you send a PeopleSoft Pure Internet Architecture URL, be sure that the recipient has permissions to access that portal , node, or page.

## Specifying Miscellaneous Parameters

Access the Miscellaneous page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Miscellaneous

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Agent Miscellaneous page

**When task is unassigned** Select from the following values the action that occurs when a task that is assigned to an agent is unassigned:

- *Prompt whether to close window* (default).
- *Close the task window.*
- *Do not close the task window.*

**Trace Level** Select from the following log trace levels:

- *0 - None*
- *1 - Information*
- *2 - Debug*

If a value other than *0* is selected, a tracer window appears to display activities and events on the chat or MultiChannel Console for debugging purposes.

**Limit debug tracer log size** This check box is enabled when the value entered for Trace Level is not 0. Select this check box to enable the agent to clear the tracer log based on Number of log messages to save when cleared and Maximum number of log messages to allow.

If the check box is deselected, the tracer log will not be cleared and the Number of log messages to save when cleared and Maximum number of log messages to allow will be disabled.

**Number of log messages to save when cleared** Specify the minimum number of recent tracer log messages that should be maintained in the tracer window.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Maximum number of log messages to allow** Specify the maximum number of tracer log messages that will be maintained in the tracer window.

---

**Note.** Number of log messages to save when cleared and Maximum number of log messages to allow fields are required if the Limit debug tracer log size check box is selected.

---

### *Limit Debug Tracer Log Size Example*

This table lists the values entered on the Miscellaneous page:

| *Field* | *Value* |
|---|---|
| Trace Level | *2 - Debug* |
| Limit debug tracer log size | Selected |
| Number of log messages to save when cleared | *25* |
| Maximum number of log messages to allow | *100* |

Based on these values, the first 75 messages will be cleared from the tracer window after 100 messages are logged. It will retain the most recent 25 messages for the agents reference. This process will repeat for every 100 messages that are logged in the tracer. The maximum number of messages is the tracer window at any one point in time is 100.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 9

# Administering Queues, Logs, and Tasks

This chapter discusses how to:

- Administer physical queues.
- View queue server, queue, and agent states.
- View broadcast, chat, and event logs.
- Administer overflow and escalated tasks.

## Administering Physical Queues

To administer physical queues, use the Physical Queues (MCF_ACCPT_TASK_CMP) component.

This section discusses how to:

- Move agents between physical queues.
- Move queues.
- Balance queues.

### Pages Used to Administer Physical Queues

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Move agent | MCF_DEMO_AGENTQ | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Physical Queues, Move agent | Move an agent to a new physical queue. |
| Move queue | MCF_DEMO_QUEUEQ | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Physical Queues, Move queue | Move agents and tasks from one physical queue to another on the same logical queue. |

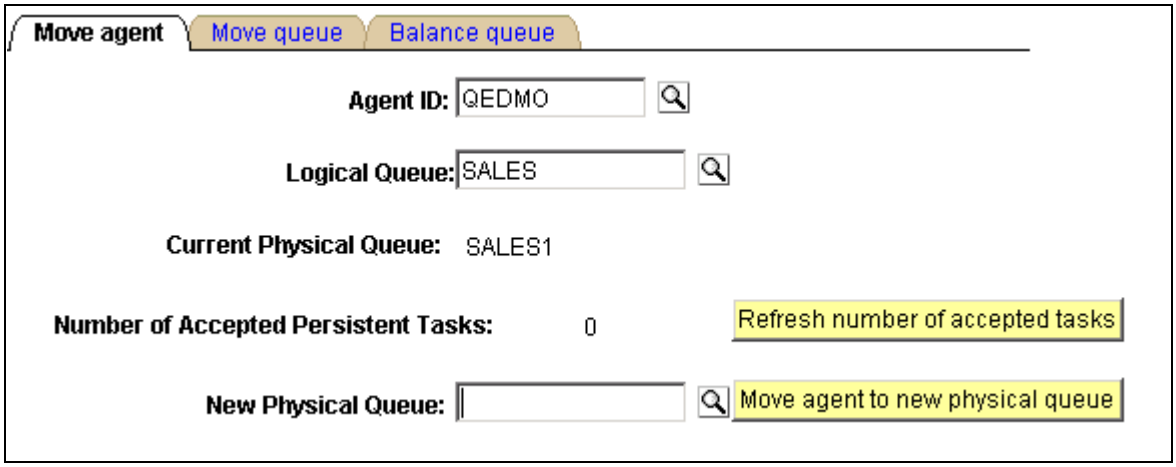
Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Balance queue | MCF_DEMO_Q_BALANCE | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Physical Queues, Balance queue | Balance agent and task assignments over active physical queues. |

## Moving Agents Between Physical Queues

Access the Move agent page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Physical Queues, Move agent




The Move agent page, which displays the Current Physical Queue and the Number of Accepted Persistent Tasks, has the editable fields Agent ID, Logical Queue, Current Physical Queue, and New Physical Queue

You can move an agent, and any open persistent tasks that are associated with that agent, from one physical queue in one cluster to another physical queue in another cluster on the same logical queue.

**Logical Queue** Select the logical queue within which the agent is to be moved.

**Number of Accepted Persistent Tasks** Displays the number of persistent tasks this agent has accepted on this physical queue.

This number is updated and displayed when you select the logical queue.

**Refresh number of accepted tasks** Click to update the number of persistent tasks accepted by this agent on the current physical queue.

**New Physical Queue** Select the new physical queue to which this agent and the persistent tasks accepted by this agent will be assigned.

**Move agent to new physical queue** Click to perform the action.

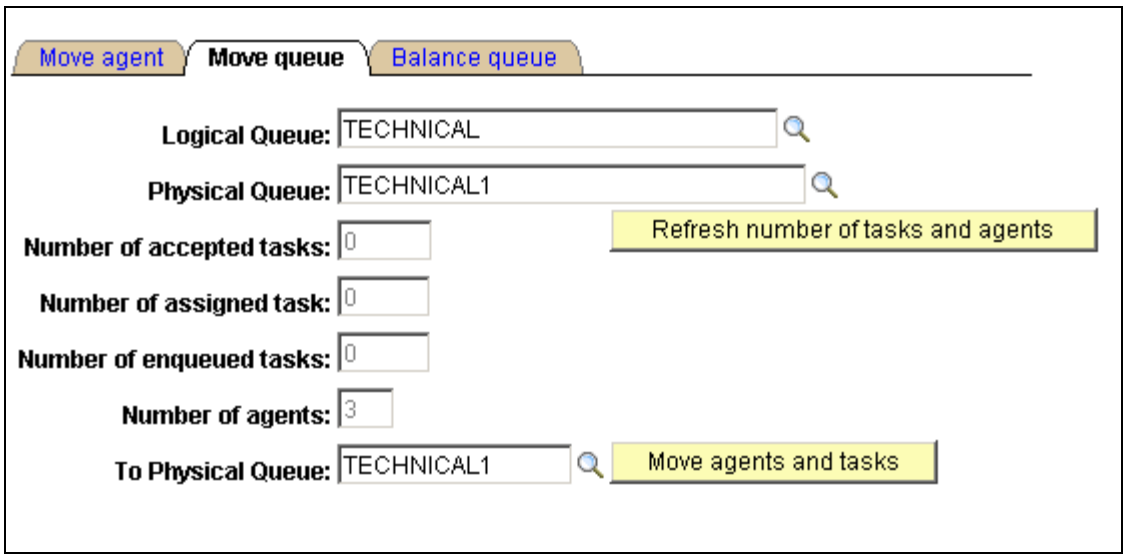

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Note.** Ongoing chat sessions, which are not persistent tasks, are not affected by the move agent action.

## Moving Queues

Access the Move queue page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Physical Queues, Move queue




The Move queue page has the editable fields Logical Queue, Physical Queue, Number of Accepted Tasks, Number of Assigned Tasks, Number of Enqueued Tasks, Number of Agents, and To Physical Queue

You can move all agents and their open persistent tasks from one physical queue to another physical queue within the same logical queue. For example, to delete a physical queue, move its agents and persistent tasks to another queue before deleting the first queue. Or, if the cluster serving this physical queue is overloaded, you can create another physical queue on another cluster, mark the first physical queue as inactive, create another physical queue on another cluster and move agents and persistent tasks from the inactive physical queue to the new physical queue.

| | |
|---|---|
| **Logical Queue** | Select the logical queue within which the selected physical queue's agents and persistent tasks will be moved. |
| **Physical Queue** | Select the physical queue from which agents and persistent tasks will be moved. |
| **Number of accepted tasks** and **Number of assigned tasks** | Displays the number of accepted and assigned tasks open on this physical queue. |
| **Refresh number of tasks and agents** | Click to update the number of agents and persistent tasks assigned to this queue. |
| **Number of enqueued tasks**and**Number of agents** | Displays the number of enqueued tasks and assigned agents on this physical queue. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**To Physical Queue** Select the physical queue to which the currently assigned agents and tasks will be moved.

**Move agents and tasks** Click to move the assigned agents and tasks to the specified physical queue.

**Note.** The physical queue must be inactive before moving agents and tasks. Inactivate the physical queue on the Queues page. Agents on the inactive physical queue are automatically logged off before being moved.

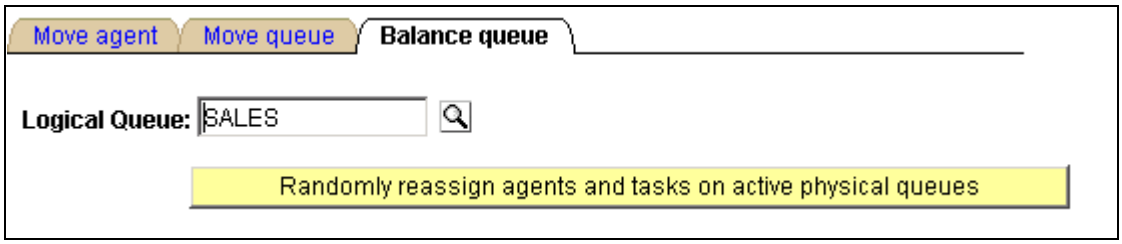
### *See Also*

Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Defining Queues, page 97

## Balancing Queues

Access the Balance queue page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Physical Queues, Balance Queue




The Balance queue page has the Logical Queue editable field

Over time, the distribution of agents and their associated skill levels, languages, and so on, across the physical queues belonging to a logical queue may change as agents are added or deleted. Rather than manually rebalancing the queue by moving individual agents, you can use the Balance queue page to randomly reassign agents and their open persistent tasks across all the active physical queues belonging to the selected logical queue.

**Logical Queue** Select the logical queue across which agents and persistent tasks will be balanced.

**Randomly reassign agents and tasks on active physical queues** Click to balance agents and tasks across the active physical queues for the selected logical queue.

This action redistributes agents and tasks assigned to this logical queue across all active physical queues without regard to previous assignments. Ensure that agents assigned to the affected physical queues have shut down their MultiChannel Consoles. They are *not* logged off automatically.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Viewing Queue Server, Queue, and Agent States

To view queue server state, queue state, and agent state, use the Queue Server State (MCF_QSERVSTATE_CMP), Queue State Summary (MCF_QUEUESTATE_CMP), and Agent State Summary (MCF_AGENTSTATE_CMP) components.

This section discusses how to:

- View the queue server state.
- View the queue state summary.
- View the agent state summary.

## Pages Used to View Queue Server, Queue, and Agent States

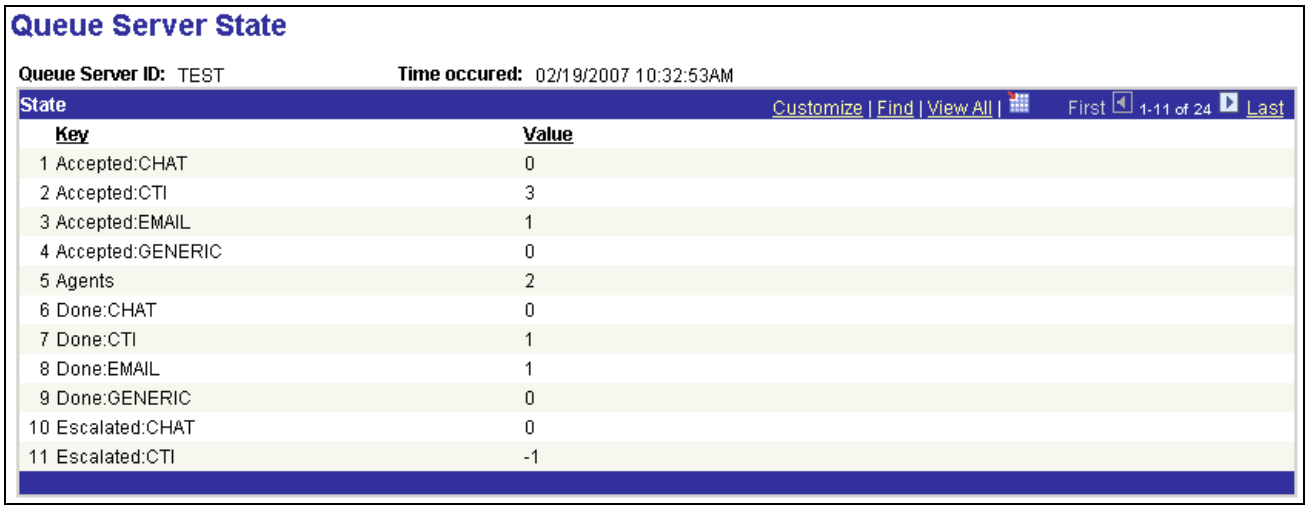
| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Queue Server State | MCFQSERVSTATE_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Queue Server State | Examine the state of a queue server at a particular time. |
| Queue State Summary | MCFQUEUESTATE_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Queue State Summary | Display the state of a queue. |
| Agent State Summary | MCFAGENTSTATE_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent State Summary | View the agent state. |

## Viewing the Queue Server State

Access the Queue Server State page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Queue Server State

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Queue Server State page displaying the Queue Server ID, Time Occurred, Key, and Value fields.

The Queue Server State page displays cumulative diagnostic totals for the selected queue server, at the selected time, across all physical queues serviced by this queue server. Cumulative totals are reset after each state dump.

This state information comprises statistics to measure load and throughput that can be analyzed and used for tuning the system. For example, if counts are high, consider adding another physical queue to balance the load.

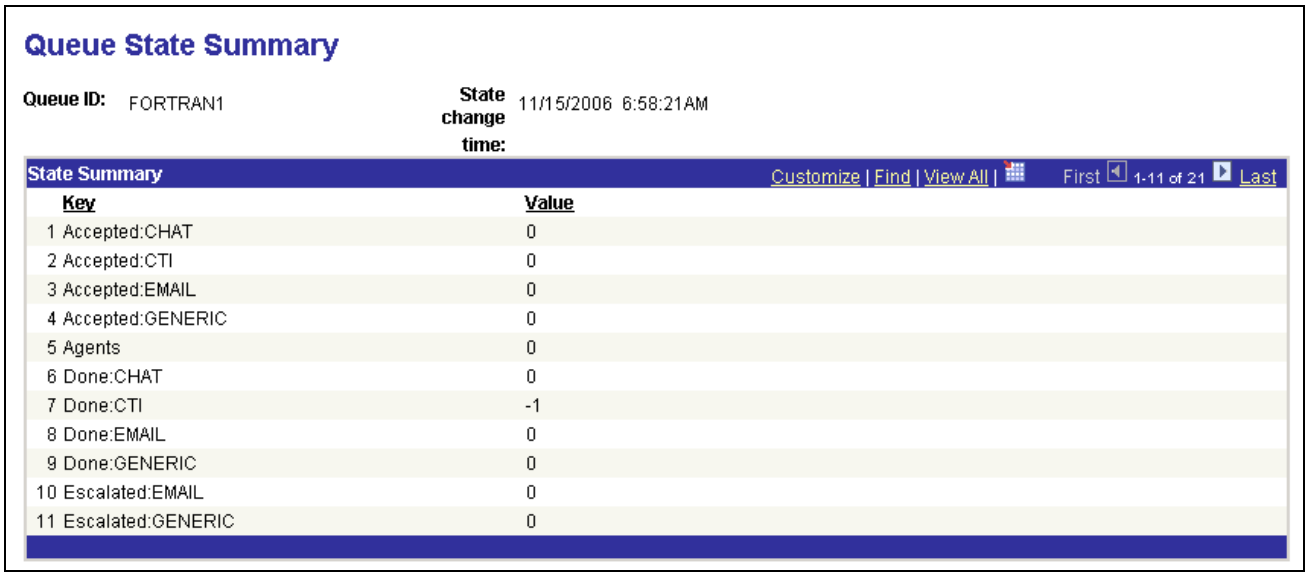
The state stamps are inserted for every primary queue server by cluster at configurable intervals, irrespective of user activity. You configure the intervals on the Cluster Tuning page.

## Viewing the Queue State Summary

Access the Queue State Summary page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Queue State Summary


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.



Queue State Summary

Queue ID: FORTRAN1 State change time: 11/15/2006 6:58:21AM

State Summary Customize | Find | View All | First 1-11 of 21 Last

| | Key | Value |
|---|---|---|
| 1 | Accepted:CHAT | 0 |
| 2 | Accepted:CTI | 0 |
| 3 | Accepted:EMAIL | 0 |
| 4 | Accepted:GENERIC | 0 |
| 5 | Agents | 0 |
| 6 | Done:CHAT | 0 |
| 7 | Done:CTI | -1 |
| 8 | Done:EMAIL | 0 |
| 9 | Done:GENERIC | 0 |
| 10 | Escalated:EMAIL | 0 |
| 11 | Escalated:GENERIC | 0 |

The Queue State Summary page displaying the Queue ID, State Change Time, Key, and Value fields.

The Queue State Summary page displays cumulative diagnostic totals for the selected physical queue.

This state information comprises statistics to measure load and throughput that can be analyzed and used for tuning the system. For example, if counts are high for this queue, consider adding another physical queue to balance the load.

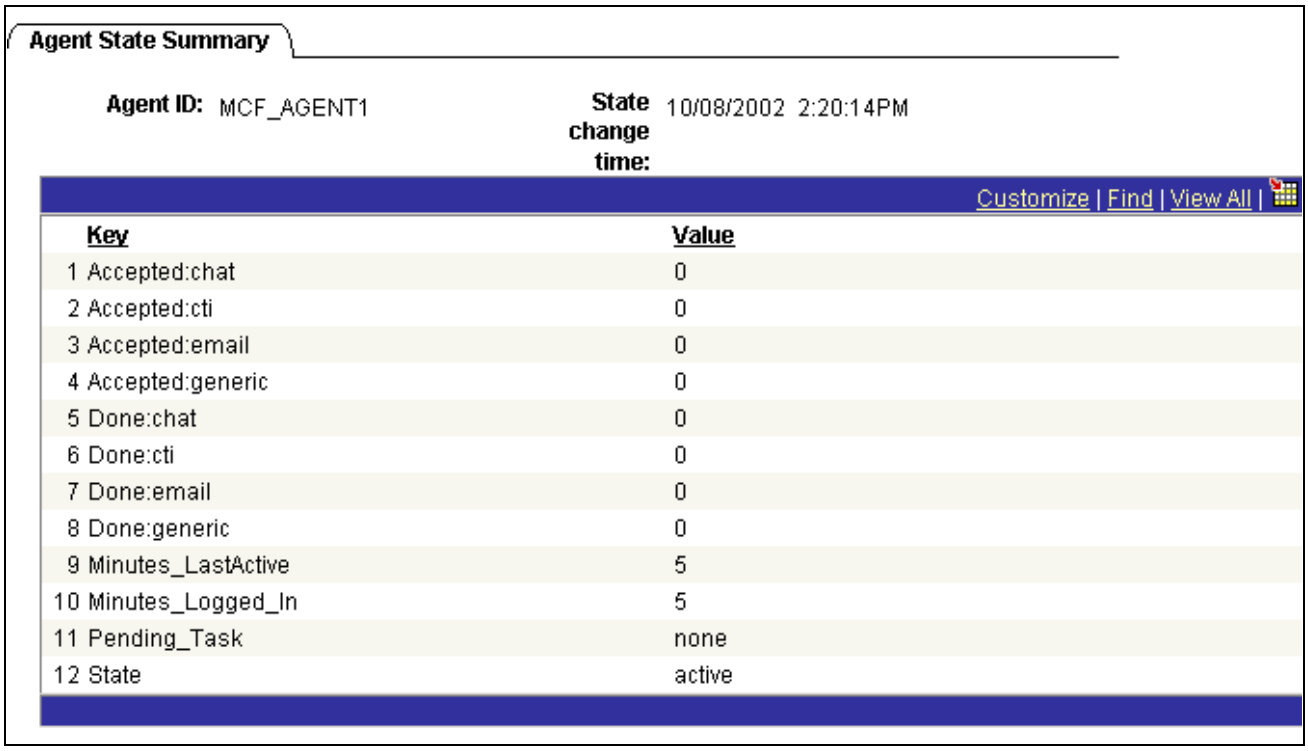
The state stamps are inserted for every physical queue at configurable intervals, irrespective of user activity. The intervals are configured on the Cluster Tuning page.

## Viewing the Agent State Summary

Access the Agent State Summary page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent State Summary

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.



Agent State Summary

Agent ID: MCF_AGENT1 State change time: 10/08/2002 2:20:14PM

Customize | Find | View All |

| | Key | Value |
|---|---|---|
| 1 | Accepted:chat | 0 |
| 2 | Accepted:cti | 0 |
| 3 | Accepted:email | 0 |
| 4 | Accepted:generic | 0 |
| 5 | Done:chat | 0 |
| 6 | Done:cti | 0 |
| 7 | Done:email | 0 |
| 8 | Done:generic | 0 |
| 9 | Minutes_LastActive | 5 |
| 10 | Minutes_Logged_In | 5 |
| 11 | Pending_Task | none |
| 12 | State | active |

The Agent State Summary page displaying the Agent ID, State Change Time, Key, and Value fields.

The Agent State Summary page displays cumulative totals for the selected agent.

This state information comprises statistics to measure agent performance and status that can be analyzed and used for performance evaluation.

The state stamps are inserted for every agent at configurable intervals, irrespective of user activity. State is only recorded for agents currently logged on. The intervals are configured on the Cluster Tuning page by setting the dumpinterval. Enable agent logging by setting dumpagent to *yes.*

An agent state of *Active* indicates the agent is available; *Inactive* indicates the agent is not available.

**Note.** On a busy system, recording state information frequently may slow performance.

### *See Also*

Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Tuning Cluster Parameters, page 108

## Viewing Broadcast, Chat, and Event Logs

To view broadcast, chat, and event logs, use the Broadcast Log (MCF_BCAST_LOG_CMP), Chat Log (MCF_CHAT_LOG_CMP), and Event Log (MCF_EVENTLOGL_CMP) components.

This section discusses how to:

- View broadcast logs.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- View chat logs.
- View event logs.
- View PeopleSoft MCF logs.

## Pages Used to View Broadcast, Chat, and Event Logs

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Broadcast | MCF_BCASTLOG_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Broadcast Log | View the log of broadcast events. |
| Chat | MCF_CHAT_MINLOG_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Chat Log | View the log of a specified chat session. |
| Event Log | MCF_EVENTLOG_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Event Log | View a log of events and display details. |

## Viewing Broadcast Logs

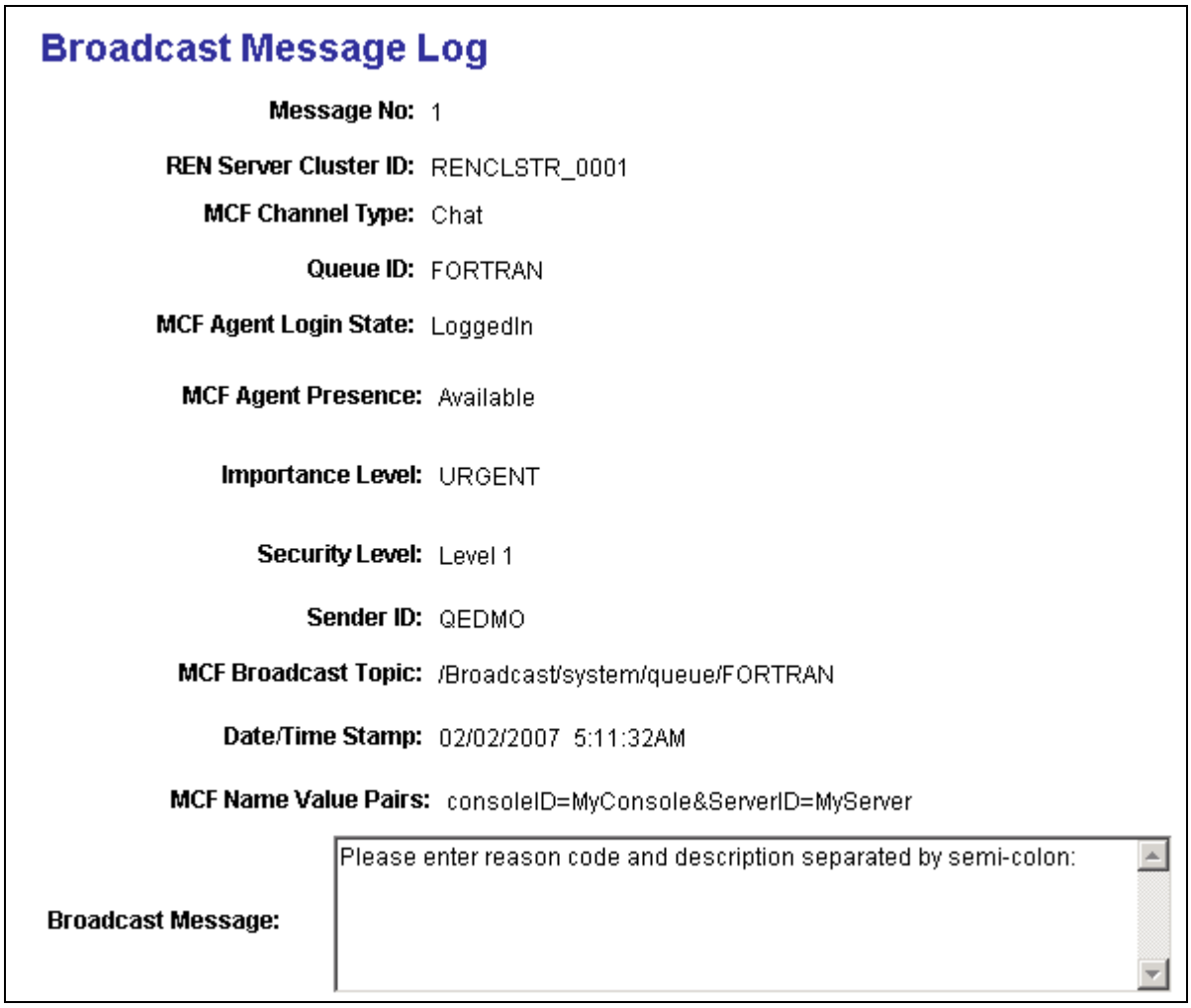
The broadcast log page displays detailed information about any broadcast messages that were sent.

Access the Broadcast Message Log page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Broadcast Log

You can also access the Broadcast Message Log page by searching by message number, queue ID, or REN cluster ID.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Broadcast Message Log page displaying the Message Number, REN Server Cluster ID, MCF Channel Type, Queue ID, MCF Agent Login state, MCF Agent Presence, Importance Level, Security Level, Sender ID, MCF Broadcast Topic, Date/Time Stamp, and the MCF Name Value Pairs. This page lets you enter a broadcast message.

---

**Note.** To view the broadcast logs, set *log_broadcast* to *Yes* on the Cluster Tuning page. For this parameter to take effect, click Refresh logging parameters on the Cluster Notify page.

---

### *See Also*

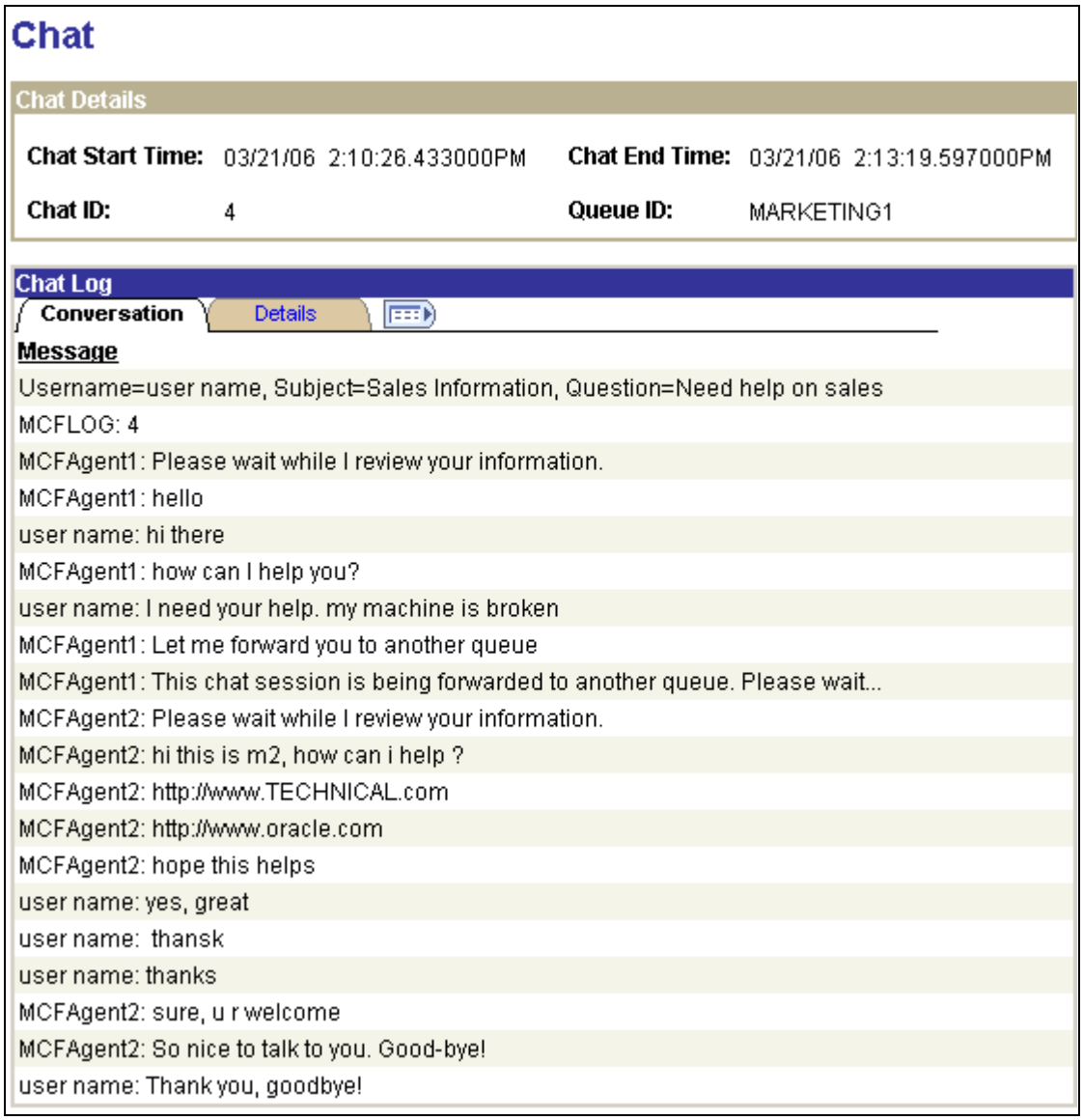
Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using PeopleSoft MCF Broadcast, page 159

## Viewing Chat Logs

Access the Chat page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Chat Log

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Conversation tab of the Chat page displaying the Chat Message along with the Chat Start Time, Chat End Time, Chat ID, and Queue ID.

If enabled, this log records the contents and events of every chat session.

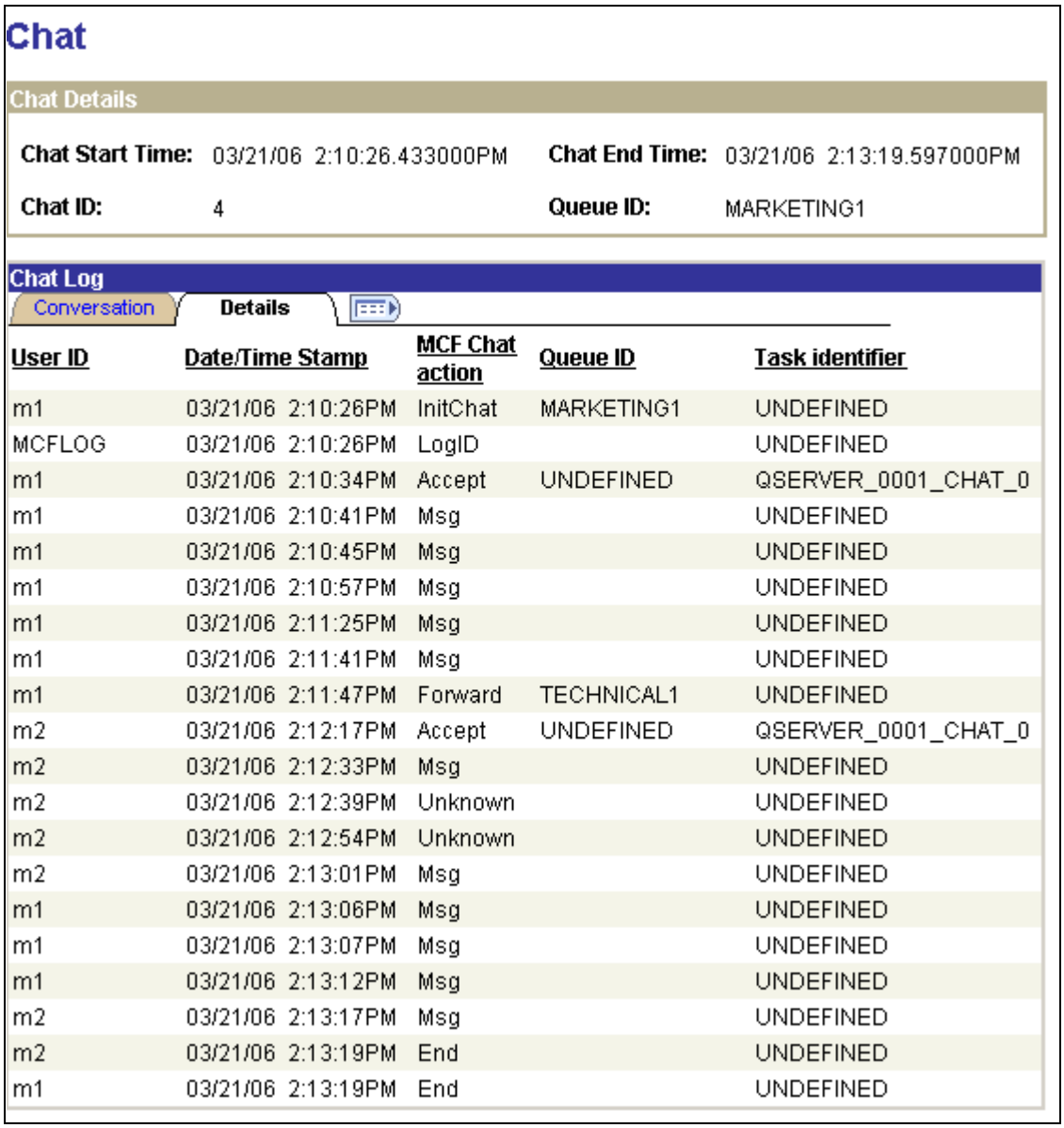
To enable chat logging, set log_chat_ses to *yes* on the Cluster Tuning page. For this parameter to take effect, click Refresh logging parameters on the Cluster Notify page.

### *Details Tab*

Access the Details tab of the Chat Page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Chat Log, Details

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.



Chat

Chat Details

Chat Start Time: 03/21/06 2:10:26.433000PM Chat End Time: 03/21/06 2:13:19.597000PM

Chat ID: 4 Queue ID: MARKETING1

Chat Log

Conversation Details

| User ID | Date/Time Stamp | MCF Chat action | Queue ID | Task identifier |
|---|---|---|---|---|
| m1 | 03/21/06 2:10:26PM | InitChat | MARKETING1 | UNDEFINED |
| MCFLOG | 03/21/06 2:10:26PM | LogID | | UNDEFINED |
| m1 | 03/21/06 2:10:34PM | Accept | UNDEFINED | QSERVER_0001_CHAT_0 |
| m1 | 03/21/06 2:10:41PM | Msg | | UNDEFINED |
| m1 | 03/21/06 2:10:45PM | Msg | | UNDEFINED |
| m1 | 03/21/06 2:10:57PM | Msg | | UNDEFINED |
| m1 | 03/21/06 2:11:25PM | Msg | | UNDEFINED |
| m1 | 03/21/06 2:11:41PM | Msg | | UNDEFINED |
| m1 | 03/21/06 2:11:47PM | Forward | TECHNICAL1 | UNDEFINED |
| m2 | 03/21/06 2:12:17PM | Accept | UNDEFINED | QSERVER_0001_CHAT_0 |
| m2 | 03/21/06 2:12:33PM | Msg | | UNDEFINED |
| m2 | 03/21/06 2:12:39PM | Unknown | | UNDEFINED |
| m2 | 03/21/06 2:12:54PM | Unknown | | UNDEFINED |
| m2 | 03/21/06 2:13:01PM | Msg | | UNDEFINED |
| m1 | 03/21/06 2:13:06PM | Msg | | UNDEFINED |
| m1 | 03/21/06 2:13:07PM | Msg | | UNDEFINED |
| m1 | 03/21/06 2:13:12PM | Msg | | UNDEFINED |
| m2 | 03/21/06 2:13:17PM | Msg | | UNDEFINED |
| m2 | 03/21/06 2:13:19PM | End | | UNDEFINED |
| m1 | 03/21/06 2:13:19PM | End | | UNDEFINED |

The Details tab of the Chat Page showing the User ID, Date, Time Stamp, MCF Chat Action, Queue ID and Task Identifier

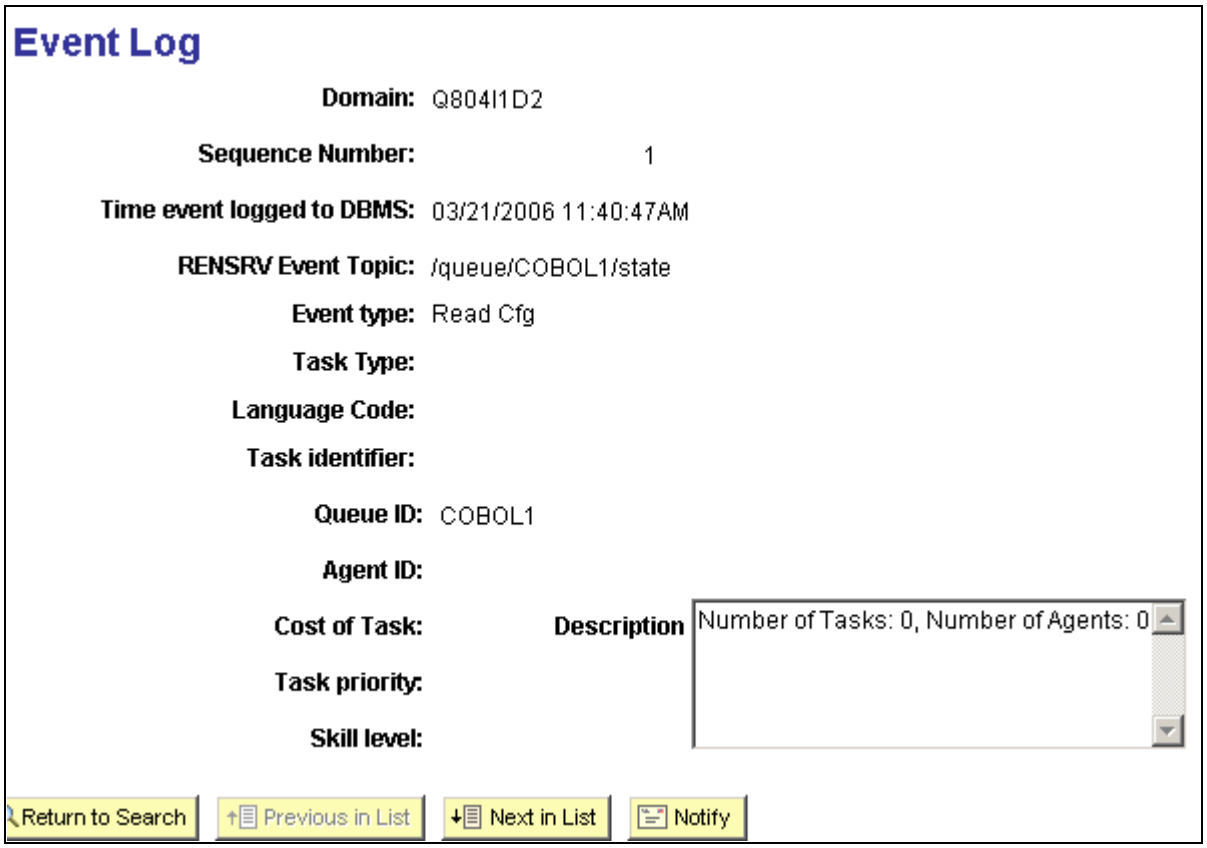
The Details tab displays detailed information about the selected chat conversation.

## Viewing Event Logs

Access the Event Log page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Event Log


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Event Log page displaying the Domain, Sequence Number, Time vent Logged to DBMS, RENSRV Event Topic, Event Type, Task Type, Language Code, Task Identifier, Queue ID, Agent ID, Cost of Task, Task Priority and Skill Level fields. This page allows you to enter a description.

| | |
|---|---|
| **Domain** | The application server domain on which this event occurred. |
| **Time event logged to DBMS** | The time that this event was recorded in the database. |
| **Event Type** | The type of event, as described in the table that follows. |
| **Task Type** | The type of task for this event: chat, email, or generic. CTI events are logged in the CTI event log. |

The event log records PeopleSoft MultiChannel Framework events sent to the real-time event notification (REN) server, excluding chat content (which is logged in the chat log). The event log can be used for debugging as well as for system monitoring. For example, you can determine when agents log in and log out, or when the queue server was first notified of newly enqueued events.

Data displayed in the event log depends on the event type. Not all fields apply to every event.

To enable logging of state broadcast events, set *logDMPQ* to *yes* on the Cluster Tuning page. For this parameter to take effect, click Refresh logging parameters on the Cluster Notify page.

Time is displayed and searched on in the format MM/DD/YYYY HH:MM:SSA/PM.

The following table lists possible event types:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Name* | *Translate Table Value* | *REN MultiChannel Framework Topic* | *Description* |
|---|---|---|---|
| Accepted | ACPD | /agent/<agentID>/accepted Agent 's list of accepted tasks | Agent's list of accepted tasks |
| Accept | ACPT | /queue/agents/accept | Agent accepts an assigned task. |
| Bcst Admin | BCST | /queue/admin/statedump | Broadcast universal queue information |
| Contact | CNCT | /queue/contact | Real-time contact (for example, chat) |
| DB Cntct | DBCT | /queue/dbcontact | Database management system contact (such as email or generic) |
| Dump Q | DMPQ | /queue/<queueID>/state | Dump queue state information to log |
| Done | DONE | /queue/agents/dequeue | Done (dequeue) |
| Forward | FWD | /queue/agents/forward | Forward |
| Notify | NTFY | /agent/<agentID>/notify | Notify agent of assigned task. |
| Presence | PRES | /queue/agents/presence | Agent's presence change |
| Restrt Ack | RACK | /queue/agents/restartack | Restart acknowledgement |
| Read Cfg | READ | /uqsrv/reread/defaults | Reread defaults |
| Restart | RSRT | /queue/<queueID>/restart | Restart |
| Unknown | UNKN | UNKNOWN | Unknown REN server event |
| Unassign | USGN | /agent/<agentID>/unassign | Unassign |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

The following table lists event logs:

| *Name* | *Translate Table Value* | *REN MultiChannel Framework Topic* | *Required Argument Value* | *Meaning* |
|---|---|---|---|---|
| Abandon | ABAN | /chat/<userID>/<chatID> | ps_type=abandon | Abandoned chat session |
| End | END | /chat/<userID>/<chatID> | ps_type=end | End chat session |
| Login | LGIN | /queue/agents/loginstate | ps_state=login | Log on |
| Logout | LGOT | /queue/agents/loginstate | ps_state=logout | Log out |
| Message | MSG | /chat/<userID>/<chatID> | ps_type=msg | Message |
| Push URL | PUSH | /chat/<userID>/<chatID> | ps_type=pushurl | Push URL |
| Timeout | TOUT | /chat/<userID>/<chatID> | ps_type=timeout | Timeout chat session |

## Viewing PeopleSoft MCF Logs

Diagnostic PSUQSRV and PSMCFLOG traces are written to the log directory of each application server domain. The trace level is determined by the LogFence setting in the domain configuration. You can set the LogFence parameter with PSADMIN configuration of the application server domain. The following table lists LogFence setting values:

| *LogFence Setting* | *Tracing Level* |
|---|---|
| 1 | Fatal errors |
| 2 | Errors |
| 3 | Warnings |
| 4 | Level 1 Diagnostic: Logs the queues that the universal queue is servicing, logs new queues that are added, and logs agent logon and logout. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *LogFence Setting* | *Tracing Level* |
|---|---|
| 5 | Level 2 Diagnostic: Logs most debugging information, except periodic events (such as timer check and heartbeat). |
| 6 | Level 3 Diagnostic: Logs everything, including periodic events. |

# Administering Overflow and Escalated Tasks

To administer overflow and escalated tasks, use the Overflow Administration (MCF_OVERFLOWL_CMP) and Escalation Administration (MCF_ESCAL_CMP) components.

This section discusses how to:

- Administer overflow tasks.
- Administer escalated tasks.

## Pages Used to Administer Overflow and Escalated Tasks

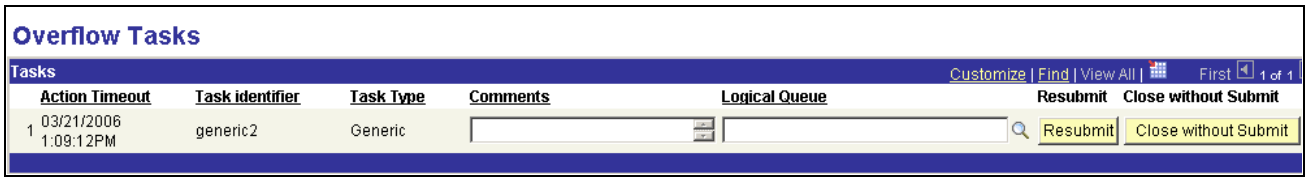
| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Overflow tasks | MCF_OVERFLOWL_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Overflow Admin (overflow administration) | View a list of overflow events. |
| Escalation Admin (escalation administration) | MCF_ESCAL_PG | PeopleTools, MultiChannel Framework, Universal Queue, Administration, Escalation Admin (escalation administration) | View a list of escalated events. |

## Administering Overflow Tasks

Access the Overflow Tasks page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Overflow Admin (overflow administration)


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Overflow Tasks page displaying the Action Timeout, Task Identifier, and Task Type fields. You can specify Comments and Logical Queue, and you can choose to Resubmit or Close without Submit on this page.

Use this page to manage tasks that could not be assigned to an agent within the specified overflow timeout.

**Action Timeout** Displays the time at which the overflow occurred.

**Task Identifier** Displays the task identifier.

**Task Type** Types include chat, email, generic, and voice (not supported).

**Comments** Enter optional text commentary about the resolution of the task.

For example, note that the customer sent follow-up email that was answered by an agent.

**Logical Queue** Select a logical queue to which to resubmit this task.

**Resubmit** Click to send the task to the specified logical queue to retry assignment.

Only persistent tasks can be resubmitted; chat cannot be resubmitted.

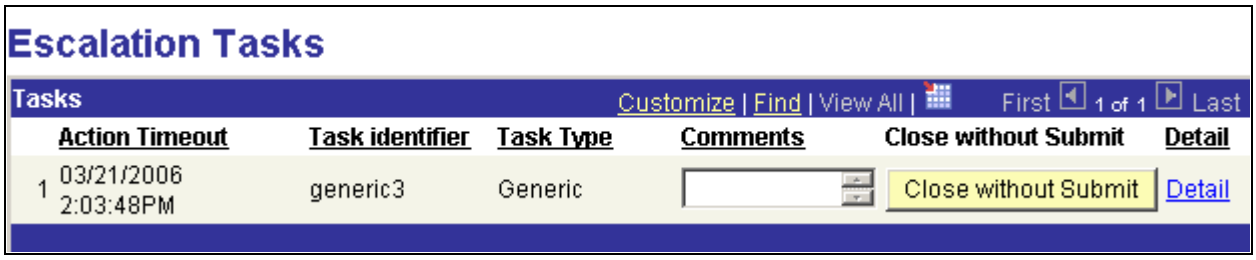
**Close without Submit** Click to close this task without sending it to retry assignment.

**Detail** Click to display additional information about the task.

## Administering Escalated Tasks

Access the Escalation Tasks page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Administration, Escalation Admin (escalation administration)




The Escalation Tasks page displaying Action Timeout, Task Identifier, Task Type, and Detail. You can enter comments and choose to Close Without Submit on this page.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Use this page to manage accepted persistent tasks that were automatically unassigned from agents because they were not closed within the specified escalation timeout. Tasks can be resubmitted or closed.

| | |
|---|---|
| **Action Timeout** | Displays when the escalation occurred. |
| **Task Identifier** | Displays the task identifier. |
| **Task Type** | Types include chat, email, generic, and voice (not currently supported). |
| **Logical Queue** | Select a logical queue to submit this task to. |
| **Resubmit** | Click to send the task to the selected logical queue to retry assignment.<br>Only persistent tasks can be resubmitted; chat cannot be resubmitted. |
| **Comments** | Enter optional text commentary about the task's resolution.<br>For example, note that the customer sent follow-up email that was answered by an agent. |
| **Detail** | Click to view additional information about this escalated task. |
| **Close without Submit** | Click to close this task without sending it back to retry assignment. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 10

# Managing Tasks and Using Chat in PeopleSoft MultiChannel Framework

This chapter discusses how to:

- Manage tasks with the MultiChannel Console.
- Communicate with customers and agents using chat.

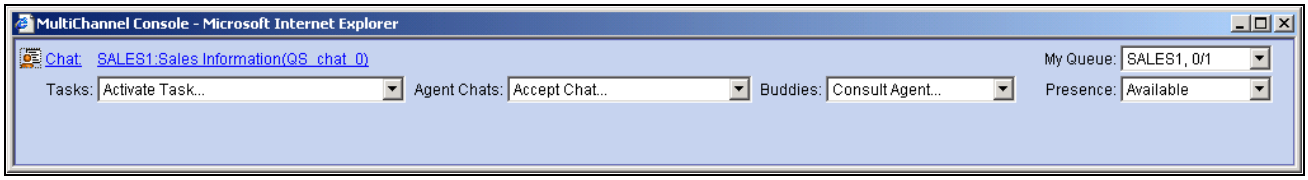
## Managing Tasks with the MultiChannel Console

Use the MultiChannel Console to accept, respond to, transfer, and complete tasks and to initiate or join chat sessions.

This section discusses how to use the MultiChannel Console to work with tasks.

### Using the MultiChannel Console to Work with Tasks

If your user ID includes security permissions for PeopleSoft MultiChannel Framework, the MultiChannel Console navigation tool appears in the universal navigation header. To access the console, click the tool.




The MultiChannel Console navigation tool having the following editable fields: Tasks, Agent Chats, Buddies, My Queue, and Presence.

You cannot launch two MultiChannel Consoles on the same workstation.

You can launch consoles on different workstations using the same user ID, but you cannot log on to physical queues served by the same MCF cluster.

When the MultiChannel Console is launched, it attempts to connect to the real-time event notification (REN) server associated with the last physical queue listed in the My Queue drop-down list box. If that REN server is not running, the agent receives an error message. If one or more of the agent's physical queues are associated with a running REN server, the agent can log on by selecting another queue.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Newly assigned tasks appear as a link above the Tasks drop-down list box, and display a flashing icon. Click the link to accept the task. Accepted tasks appear in the Tasks drop-down list box.

Collaborative chat requests (agent-to-agent chats) may appear as a link above the Agent Chats drop-down list box, but all collaborative chat requests (accepted or not) appear in the Agent Chats drop-down list box.

Configure the size and initial location of the MultiChannel Console on the agents Window Config (window configuration) page.

---

**Note.** If you use web content zones in Microsoft Internet Explorer security options, your PeopleSoft Pure Internet Architecture web server address and your REN server address must be in the same security zone.

If you use Secure Sockets Layer (SSL) security, you may receive a security warning message when your console first opens. You can accept the warning message without compromising SSL security.

---

**Tasks**

After you accept an assigned task, the task appears here.

Activate a task to work on by selecting it. Activating a task brings the window associated with that task to the foreground. If the associated task window is not running, it is launched by the console.

A task is removed from the task list when you mark the task as done using one of the following methods:

- For email and generic tasks, the application page of the task may include a Done button; refer to the application's documentation.
- For chat, the task is complete when the chat dialog ends.

**Agent Chats**

Displays a list of your collaborative chats, both accepted and requested, including a label indicating if the chat is inbound or outbound. Select a chat to activate it.

If you are invited to join in a chat session (conference), the conference chat is added to the list. After you accept the conference, the conference is added to the Tasks drop-down list box and removed from the Agent Chats drop-down list box. The chat is removed from the list when the chat dialog ends.

**Buddies**

Includes agents that are identified as buddies in your agent configuration. Add buddies on the Buddy List page of the Agent component.

To initiate a collaborative chat, select an agent. A chat dialog box displays.

---

**Note.** You can only initiate a collaborative chat with an agent who appears on your buddy list. However, if you appear on another agent's buddy list, that agent can initiate a collaborative chat with you, even if that agent is not one of your buddies.

---

**My Queue**

Select a queue to log on to that queue.

The drop-down list box includes the physical queues that you can access.

To change queues, select a new queue. You are logged off from the old queue and logged on to the selected queue.

Add or delete queues on the Queues page of the Agents component.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Presence**

Displays your current presence and enables selection of a new presence. Select from:

- *Active*
- *Inactive*

Customize presence labels to more closely track agent activity. Edit your presence options on the Presence page of the Agent component.

If you select an inactive presence, an icon appears in the upper-left corner of the MultiChannel Console. Changes in agent presence are logged and include the customized presence description.

Presence status persists for eight hours from the time of its last change.

You can use hot key combinations, described in the following table, to navigate between MultiChannel Console fields:

| ***Hot Key Combination*** | ***Opens*** |
|---|---|
| ALT + A | Agent chats |
| ALT + B | Buddies |
| ALT + P | Presence |
| ALT + Q | My Queues |
| ALT + T | Tasks |

# Communicating with Customers and Agents Using Chat

This section discusses how to:

- Use the agent chat window.
- Use the customer chat window.

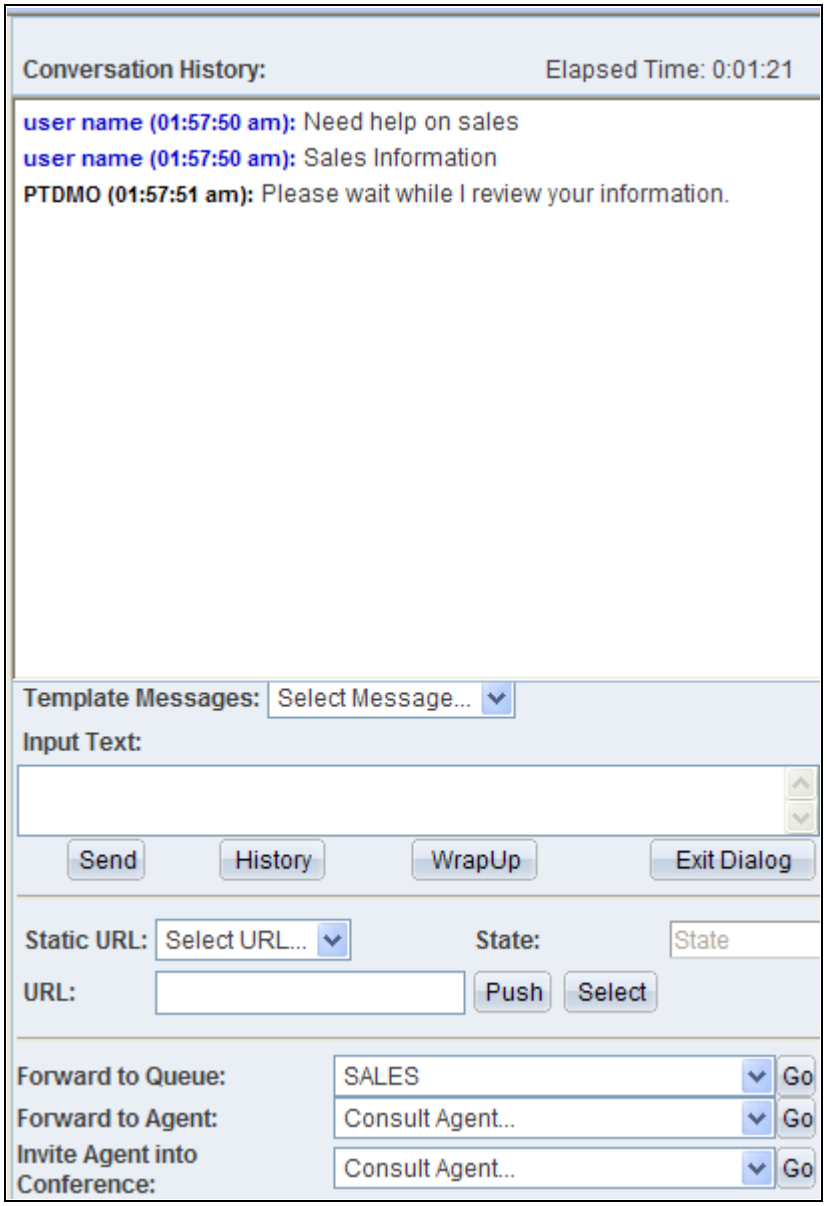
## Using the Agent Chat Window

When you accept a chat task, a chat window opens.

The format of the agent chat window and the contents of its right pane are delivered as part of PeopleSoft MultiChannel Framework. The content of the left pane is determined by application developers, and is passed as a parameter to the InitChat() built-in function.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

The following example shows an agent chat window:




Agent Chat window

Configure the size and initial location of the agent chat window on the agent's Window Config (window configuration) page.

**Conversation History** Lists progress of the chat, line by line.

If chat logging is enabled, the conversation is recorded in the database chat log by the MCF log server. View the chat log on the Chat Log page.

If accessibility features are not turned off in the My Personalizations component, an additional text box appears below the conversation history. The most recent customer input appears in the secondary text box, which can be read by screen reading software.

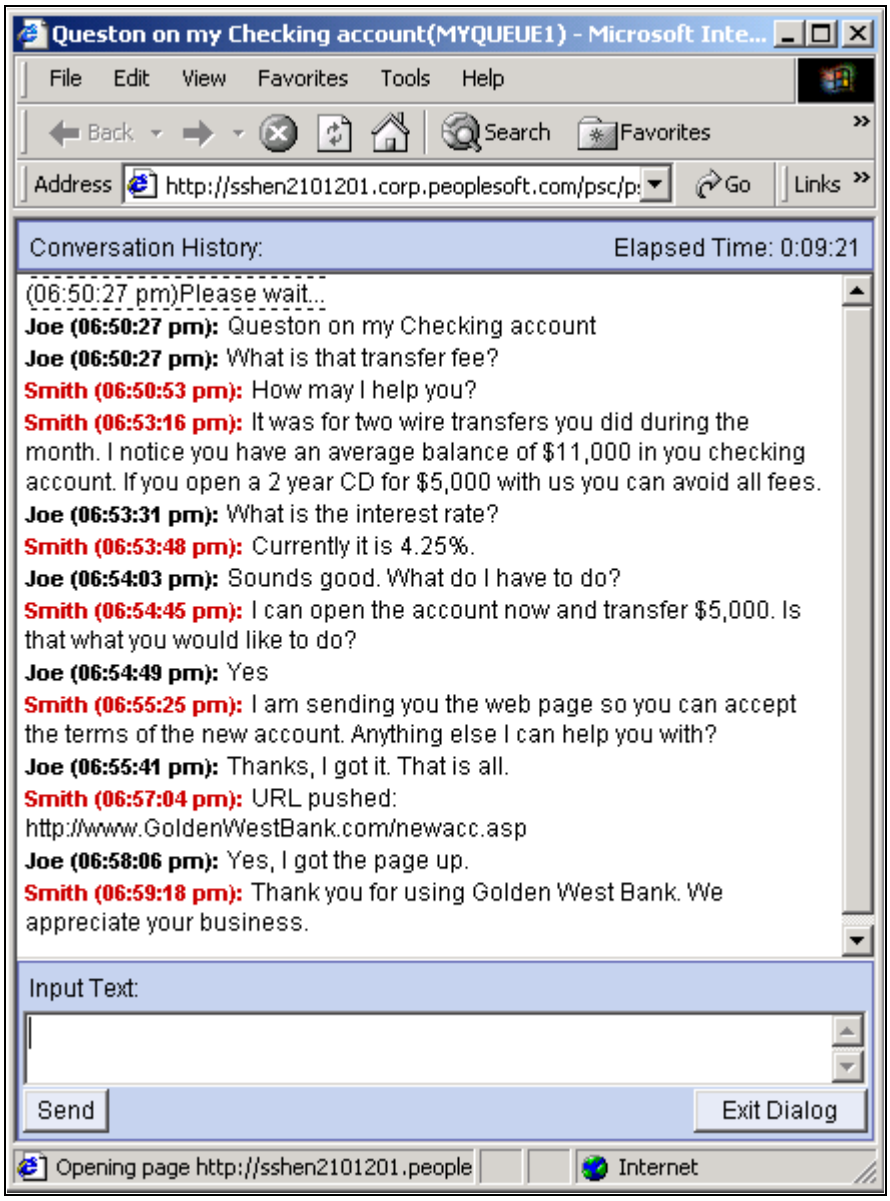

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| | |
|---|---|
| **Template Messages** | Send the customer a standard message by selecting one from the list.<br>The message text appears in the Input Text text box. Click Send to send the message.<br>Edit template messages for each queue on the queue Chat Responses page. |
| **Input Text** | To respond to the customer, enter text, and then click Send or press Enter.<br>The maximum size of the text that can be sent at one time is 4096 bytes (4 kilobytes). |
| **Send** | Click to send the contents of the Input Text text box. |
| **Exit Dialog** | Click to end the chat.<br>The chat window remains open, enabling you to return to the page for follow-up work after the customer exits the dialog. |
| **Static URL** and **Push** | To send a static URL to the customer, select a URL name, then click Push.<br>When a URL is pushed to a customer, a browser window for that URL is launched on the customer's workstation. |
| **URL** | Displays static and grabbed URLs. |
| **Select** | Click to launch an application page, from which a URL can be returned to populate the URL field.<br>The format of the URL wizard page that appears when you click Grab is defined by application developers. PeopleSoft supplies a sample URL wizard page.<br>See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the URL Wizard, page 172. |
| **Forward to Queue** | To forward the current chat to another queue, select the queue.<br>You can only forward to another physical queue, and that physical queue must be served by the same MCF cluster as the physical queue that assigned the chat. |
| **Invite Buddy** | To request that another agent join the chat, select the agent and click Go.<br>The buddy must be logged on to a physical queue that is served by the same MCF cluster as the physical queue that sent you this chat. Chat conferencing is only supported on customer-initiated chat sessions. |

## Using the Customer Chat Window

When a customer initiates a chat, a chat window appears:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




A Customer chat window showing the Conversation History and the Input Text

The format and content of the customer chat window is delivered as part of PeopleSoft MultiChannel Framework. The customer chat window includes a conversation history text box, input text box, and Send and Exit Dialog buttons.

If accessibility features are not turned off in the My Personalizations component, an additional text box appears below the conversation history. The most recent agent input appears in the secondary text box, which can be read by screen reading software.

**Input Text** The customer enters text here.

The maximum size of the text that can be sent at one time is 4096 bytes (4 kilobytes).

**Send** Click to send the input text to the agent.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Exit Dialog** Click to end the chat and close the chat window.

---

**Note.** The agent collaborative chat window is substantially the same as the customer chat window.

---

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 11

# Using PeopleSoft MCF Broadcast and Working with Sample Pages

This chapter discusses how to:

- Configure PeopleSoft MCF Broadcast.
- Work with sample pages.
- Use and demonstrate the JavaScript MultiChannel API (JSMCAPI).
- Use PeopleCode built-in functions.
- Use Universal Queue classes.

## Using PeopleSoft MCF Broadcast

Use the broadcast function to broadcast a notification message. This function is typically used by a supervisor to send a notification message to specific recipients based on the parameters that are provided by the sender. Broadcast notifications can be sent system-wide, cluster-wide, queue-based, task-based, or activity-based.

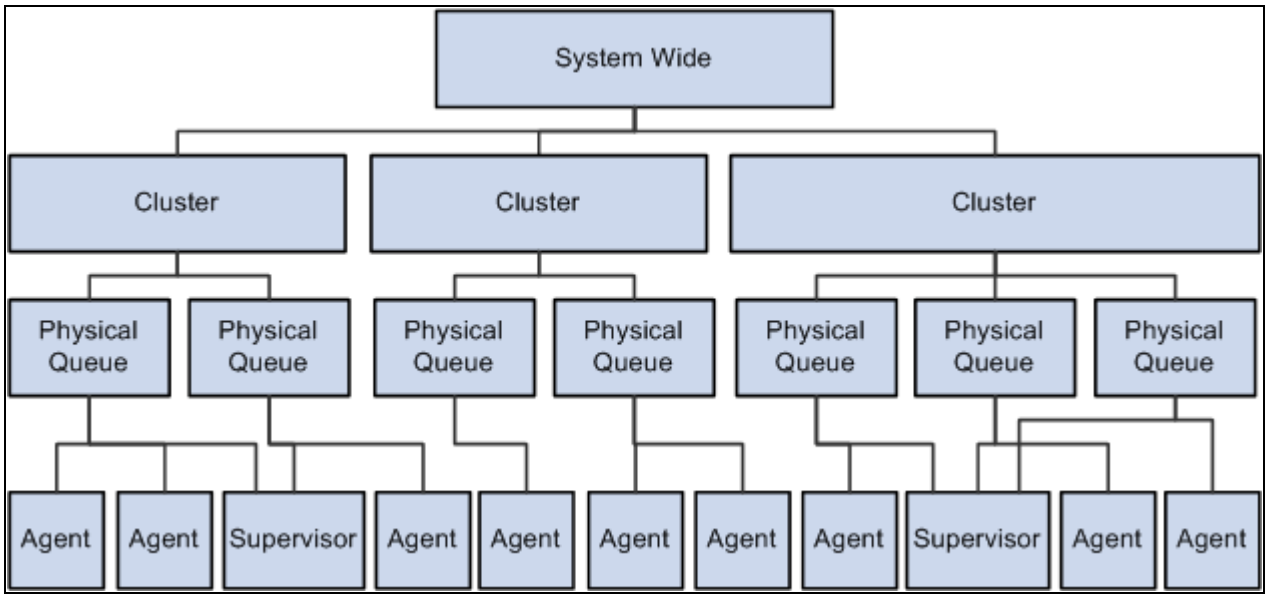
System-wide broadcast notifications can be sent only using the PeopleCode API to all logical queues on the system. The same notification using JSMCAPI is sent to all the physical queues on the cluster.

Cluster-wide broadcast enables a user to broadcast to an entire audience that is logged on or subscribed to this broadcast topic on a particular cluster. A cluster broadcast can be configured to target a specific audience by specifying a particular queue, a channel, or an agent log state in the broadcast call.

A queue-based broadcast can send a notification to all agents on a particular queue. A channel-type broadcast is received by agents serving a particular channel. An agent working on multiple channels can receive broadcast that is meant for all those channels. An agent login state or activity-based broadcast allows a broadcast notification to be sent to all agents in a particular state.

The following diagram illustrates a cluster-wide broadcast:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Cluster-wide broadcast

This section provides an overview of JSMCAPI broadcast and discusses how to:

- Implement MCF broadcast.
- Configure JSMCAPI Broadcast using MCF Supervisor console.
- Configure PeopleCode broadcast.
- View broadcast logs.

## Pages Used to Configure PeopleSoft MCF Broadcast

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| JSMCAPI Broadcast | MCF_BROADCAST | PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, JSMCAPI Broadcast | Configure JSMCAPI Broadcast using MCF Supervisor Console. |
| PCodeBroadcast | MCF_DEMO_BROADCAST | PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, PCodeBroadcast | Send a broadcast message using PeopleCode broadcast. |

## Understanding JSMCAPI Broadcast

A JSMCAPI console primarily operates on physical queues. Whenever an agent or a supervisor uses a JSMCAPI console, the agent chooses a physical queue and logs onto it. To broadcast a message, a console typically uses this physical queue as a parameter to determine the target cluster. JSMCAPI broadcast works on the following restrictions:

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- JSMCAPI broadcast function does not provide any routing logic because it operates within a limited set of physical queues that are assigned to the agent or supervisor.

  The agent can choose any physical queue from the assigned queue list without logging onto any particular physical queue.

- The supervisors or the users of JSMCAPI broadcast can use only the physical queues on their list to send a broadcast notification.

- JSMCAPI works within the confine of a single cluster and does not have any knowledge or access to other clusters in the system.

## Implementing MCF Broadcast

To enable you to implement broadcast, MultiChannel Framework provides a PeopleCode built-in function and a JSMCAPI interface for JSMCAPI users (agents and supervisors) to send broadcast notifications.

---

**Note.** The broadcast notifications are displayed only in the third-party sample pages.

---

### *Subscribing and Publishing Broadcast*

The JSMCAPI broadcast function is a specialized publish call to the REN server. Both JSMCAPI and PeopleCode publish to physical-queue broadcast topic (such as /Broadcast/SALES1), and the subscribers always subscribe to broadcast topic for the physical queues to which they were assigned.

To subscribe, use the following function:

```
BroadcastSubscribe(cluster,queue,task,state,presence,method)
```

To publish, use the following function:

```
MCFBroadcast(cluster,queue,task,state,presence,message,securitylevel,
importancelevel, senderid, NameValuePairString
```

To unsubscribe, use the following function:

```
void broadcastUnsubscribe(type)
```

where <type> determines the type of broadcast, system-wide, queue-wide, or agent-wide.

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," broadcastSubscribe, page 453

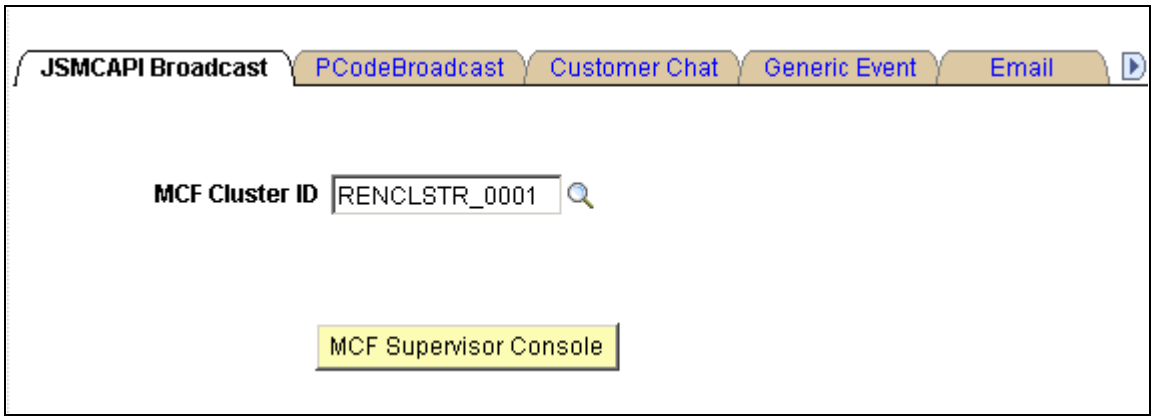
Chapter 13, "Understanding JSMCAPI Classes," broadcastUnsubscribe, page 453

## Using JSMCAPI Broadcast with MCF Supervisor Console

Access the JSMCAPI Broadcast page using the following navigation path:

PeopleTools ,MultiChannel Framework, Universal Queue, Sample Pages, JSMCAPI Broadcast

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 




The JSMCAPI Broadcast page having the MCF cluster ID field

**REN Server Cluster ID** Select the REN server cluster on which to test the MCF Supervisor console.

**Note.** The MCF Supervisor console is specific to the REN cluster Id selected.

**MCF Supervisor Console** Click to initiate the supervisor console.

**Note.** To demonstrate the MCF Supervisor console, you must have MCF servers, both queue and log servers, running and communicating with the specified REN server cluster.

After you click MCF Supervisor console, a new browser window appears that displays the sample supervisor console.

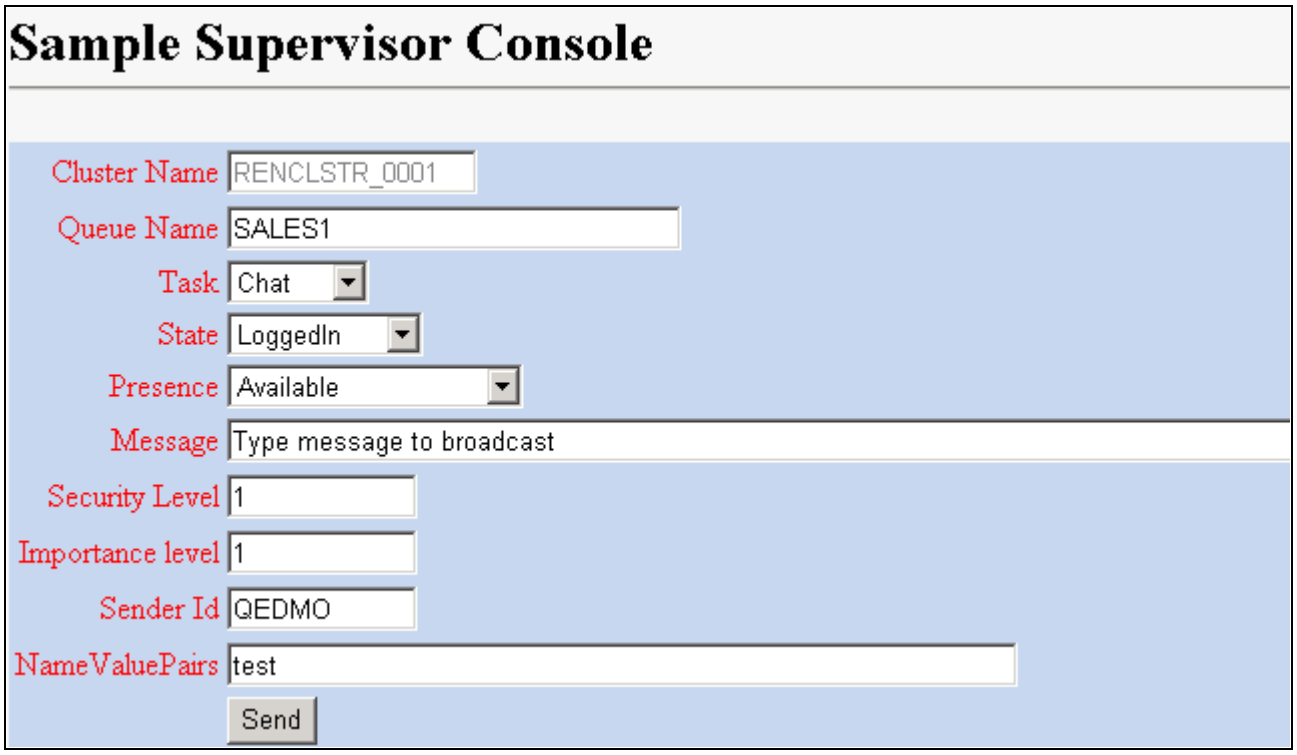
**Note.** To enable the new browser window to appear, disable any pop-up blocking software for your browser.

### *MCF Supervisor Console*

Access the MCF Supervisor console using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages. Choose an MCF Cluster ID and Click on the MCF Supervisor Console to open the MCF Supervisor Console.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The MCF Supervisor Console displaying the Cluster Name and having the following editable fields — Queue Name, Task, State, Presence, Message, Security Level, Importance Level, Sender ID, and Name Value Pairs.

**Cluster Name** Displays the REN server cluster on which to test broadcast.

**Queue Name** Enter the name of the queue that is in the specified cluster.

**Note.** Enter the queue name only when you want to broadcast the message to a particular queue.

**Task** Select the task. Values are *email,chat,voice,generic,* and *none.*

**State** Select the state of the agent.

**Note.** If the state of the agent is *LoggedIn* when the broadcast message is being sent by the supervisor, the agent receives the broadcast message using Agent console.

**Presence** Select the presence.

**Note.** If the agent is logged out, the presence should be inActive. The broadcast message is received only when the agent is currently logged in to the queue on which broadcast is sent.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Message** Enter the message that you want to broadcast.

**Note.** To view the broadcast message that is sent by the supervisor, use the Agent console sample page. Select the same REN cluster that is used by the supervisor to broadcast, and the broadcast message is displayed on the Agent console.

See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the Agent Console Page, page 192.

**Security Level** (Optional) Enter the security level.

**Importance Level** (Optional) Enter the importance level.

**Sender Id** (Optional) Enter the sender ID. Use this option only when you want the sender ID to appear as something other than the ID that actually sent the message.

**NameValue Pairs** Enter the name and value pairs to configure your data.

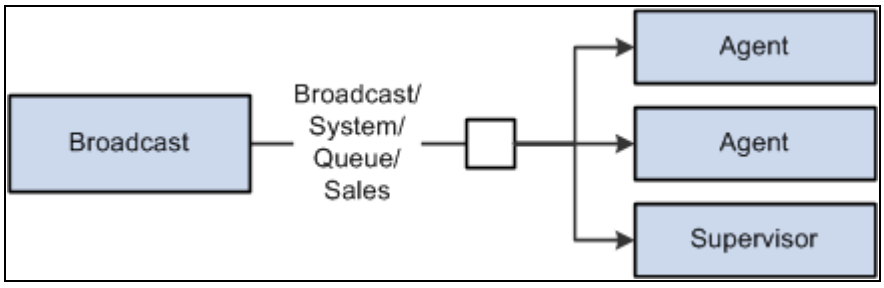
**Note.** These name-value pairs are concatenated as a string. You can define any name-value parameters for your application and send them in the name-value pair string. The JSMCAPI passes the same as a string to the console. Your particular application needs to process that string accordingly.

**Note.** For a system-wide or a cluster-wide broadcast, the broadcast message from the supervisor is sent to all agents, as well as the supervisors who are logged in to the cluster that is specified by the supervisor. The system-wide broadcast is same as cluster-wide broadcast because a JSMCAPI supervisor knows only the cluster it is operating on.

For a queue-wide broadcast, the broadcast message from the supervisor is sent to all agents who are currently logged in to the queue that is used by the supervisor to broadcast the message. This assumes that the queue is in the same cluster as the supervisor who is sending the broadcast message.

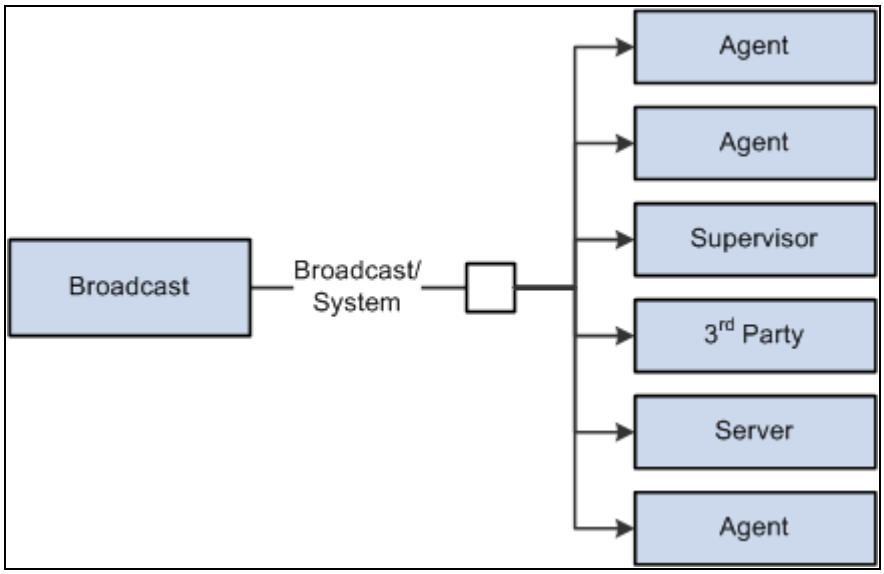
For agent-wide broadcast, the broadcast message will be sent to all agents who are currently logged to any queue in the same cluster as the supervisor who is sending the message.

The following diagrams illustrate queue-wide and cluster-wide broadcast:




Queue-wide broadcast

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Cluster-wide broadcast

The following table describes the combinations:

| ***Type of Broadcast*** | ***Cluster*** | ***Queue*** | ***Agent Login State*** | ***Agent Activity State*** | ***Broadcast Audience and Description*** |
|---|---|---|---|---|---|
| JSMCAPI | Not specified | Not specified | Not specified | Not specified | This is a cluster-wide broadcast. All agents on Cluster_1 will receive the broadcast. |
| JSMCAPI | Cluster_1 | Not specified | Not specified | Not specified | Same as the preceding. All agents on Cluster_1 will receive the broadcast. |
| JSMCAPI | Cluster_1 | SALES | Not specified | Not specified | All agents subscribing to SALES will receive the broadcast. This assumes that SALES is on Cluster_1. |

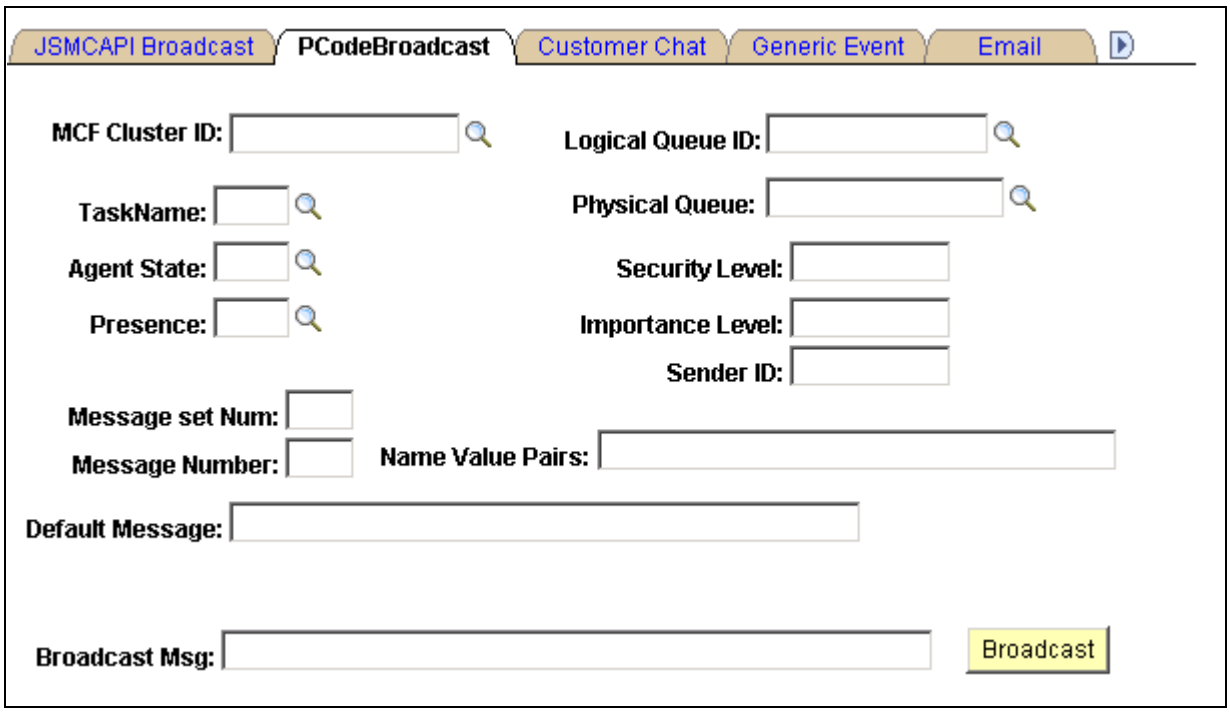
Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Type of Broadcast* | *Cluster* | *Queue* | *Agent Login State* | *Agent Activity State* | *Broadcast Audience and Description* |
|---|---|---|---|---|---|
| JSMCAPI | Cluster_1 | SALES | Logged | Not specified | All agents who are logged (active and inactive) to sales will receive the broadcast. This assumes that SALES is on Cluster_1. |

## Using PeopleCode Broadcast

Access the PCodeBroadcast page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, PCodeBroadcast




The PCodeBroadcast page having the following editable fields — MCF Cluster ID, Logical Queue ID, Task Name, Physical Queue, Agent State, Security Level, Presence, Importance Level, Sender ID, Message Sent Number, Message Number, Name Value Pairs, Default Message, and Broadcast Message

**MCF Cluster ID** Select the target cluster receiving the broadcast message.

**Logical Queue ID** Select the name of the target logical queue.

**Physical Queue ID** Select the name of the target physical queue.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**TaskName** Select the task. Select from *email, chat,voice,generic* or *none*.

**Agent State** Select the state of the agent.

**Note.** The agent receives the broadcast message only if the agent is logged in.

**Presence** Select the presence.

**Note.** If the agent is logged out, the presence should be *inActive.* All agents who are currently logged in receive the broadcast message.

**Message Set Number** Select the message set number if you want to broadcast a message from the Message Catalog.

**Message Number** Select the message number in the message set.

**Default Message** Enter the default message.

If no message is found in Message Catalog with the preceding message number, the text of the default message is used for the broadcast instead.

**Security Level** (Optional) Enter the security level.

**Importance Level** (Optional) Enter the importance level.

**Sender Id** Enter the sender ID if you want the sender that is displayed with the notification to be an ID other than the one that sent the broadcast.

**Message** Enter the message that you want to broadcast.

**Note.** To view the broadcast message that is sent by the supervisor, use the Agent console sample page. Select the same REN cluster as used by the supervisor to broadcast, and the broadcast message is displayed on the Agent console.

See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the Agent Console Page, page 192.

**NameValue Pairs** Enter the name and value pairs to configure your data.

**Note.** These name-value pairs are concatenated as a string. You can define any name-value parameters for your application and send them in the name-value pair string. The JSMCAPI passes the same as a string to the console. Your particular application needs to process that string accordingly

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Note.** For a system-wide PeopleCode Broadcast, the broadcast message from the supervisor is sent to all agents, as well as the supervisors on all active REN clusters.

For a cluster-wide PeopleCode Broadcast, the broadcast message from the supervisor is sent to all agents, as well as the supervisors on the specified REN clusters.

For a queue-wide broadcast, the broadcast message from the supervisor is sent to all agents who are currently logged in to the queue that is specified by the queue ID in all clusters if the MCF cluster ID is not specified. If the cluster ID is specified, all agents that are logged into the queue that is specified by the queue ID in the cluster that is specified by the cluster ID receive the broadcast message.

For agent-wide broadcast, if both the cluster ID and queue ID are specified, the broadcast message is sent to all agents that are currently logged into the queue that is specified by the queue ID. This assumes that the queue is in the cluster that is specified by the cluster ID. However, if only the cluster ID is mentioned, all agents that are logged and active on any queue on that cluster receive the broadcast message.

The following table describes the combinations:

| *Type of Broadcast* | *Cluster* | *Queue* | *Agent Login State* | *Agent Activity State* | *Broadcast Audience and Description* |
|---|---|---|---|---|---|
| PeopleCode | Not specified | Not specified | Not specified | Not specified | This is a true system-wide broadcast. All agents on all active REN clusters receive the broadcast. |
| PeopleCode | Cluster_1 | Not specified | Not specified | Not specified | All agents on Cluster_1 receive the broadcast. |
| PeopleCode | Cluster_1 | SALES | Not specified | Not specified | All agents subscribing to SALES receive the broadcast. This assumes that SALES is on Cluster_1. |
| PeopleCode | Not specified | SALES | Logged | Not specified | All agents that are logged (active and inactive) to SALES (all clusters with physical queues that are associated with SALES) receive the broadcast. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Type of Broadcast* | *Cluster* | *Queue* | *Agent Login State* | *Agent Activity State* | *Broadcast Audience and Description* |
|---|---|---|---|---|---|
| PeopleCode | Cluster_1 | SALES | Logged | Active | All agents that are logged into SALES and are in active state receive the broadcast. This assumes that SALES is on Cluster_1. |
| PeopleCode | Cluster_1 | Not specified | Logged | Active | All agents that are logged and active on any queue on Cluster_1 receive the broadcast. |
| PeopleCode | Cluster_1 | Not specified | Not specified | Active | All agents that are logged and active on any queue on Cluster_1 receive the broadcast. This is the same as the preceding because only agents that are logged in can be active. |

#### *See Also*

*PeopleTools 8.51 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions," MCFBroadcast

## Viewing Broadcast Logs

To view broadcast log, use the Broadcast log page.

See Chapter 9, "Administering Queues, Logs, and Tasks," Viewing Broadcast Logs, page 141.

# Working with Sample Pages

To demonstrate MCF tools and functionality, use the MCF Sample Pages (MCF_DEMO_CMP) component.

This section discusses how to:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

- Use the Customer Chat sample page.
- Use the URL wizard.
- Use the Generic Event sample page.
- Use the Generic Event window.
- Use the Email sample page.
- Use the Email window.

## Pages Used to Work with Sample Pages

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Customer Chat | MCF_DEMO_PG | PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Customer Chat | View a customer chat session and built-in functions. |
| Generic Event | MCF_DEMO_NOTIFY_PG | PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Generic Event | View a generic event notification and built-in functions. |
| Email | MCF_DEMOEM_NTFY_PG | PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Email | View email retrieval, built-in functions, and application package classes. |

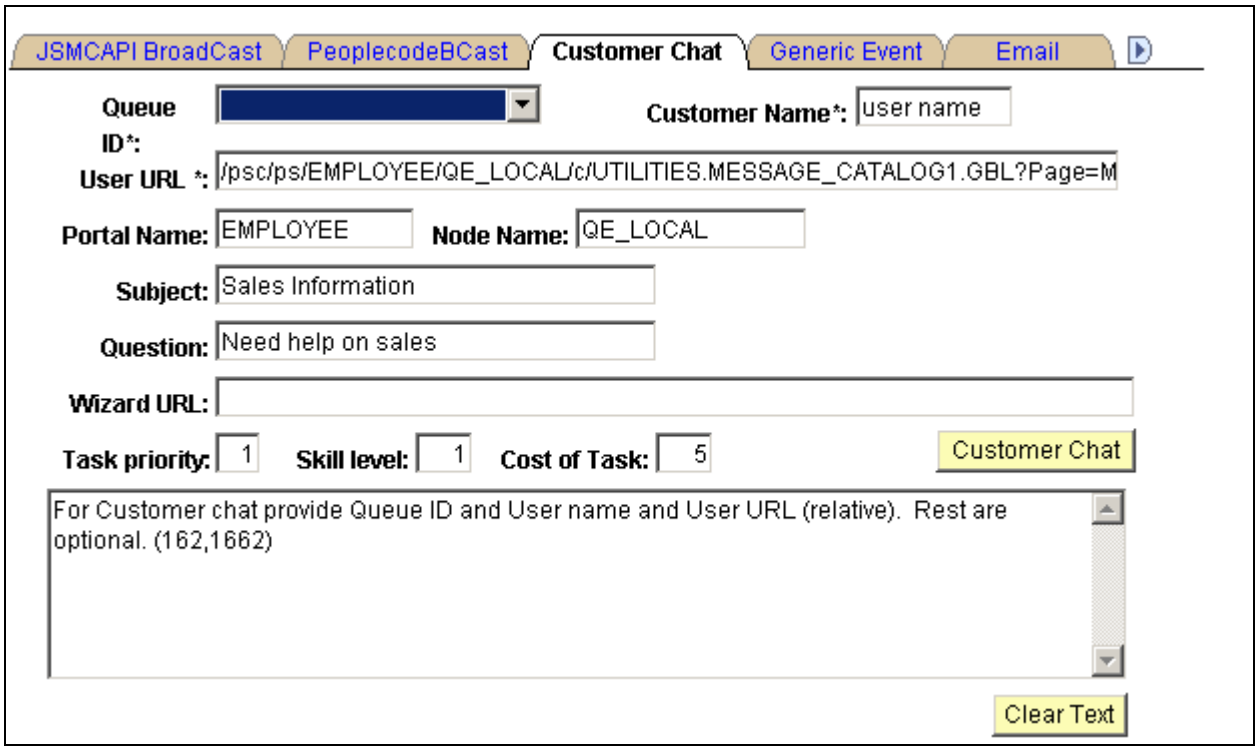
## Using the Customer Chat Sample Page

Access the Customer Chat page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Customer Chat

**Note.** This page is available using both queue server and third-party routing server.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Customer Chat page having the following editable fields — Queue ID, customer Name, User URL, Portal Name, Node Name, Subject, Question, Wizard URL, Task Priority, Skill Level, and Cost of Task

This page demonstrates the InitChat() built-in function, including required and optional parameters.

This page is not intended for production use. On a typical application page, the developer includes a Live Help or Customer Chat button that calls the InitChat() built-in function and passes required parameters, including the context of the chat request. The sample customer chat page demonstrates values that can be included in the InitChat() parameters. The user may not be prompted for any information.

To run a sample chat:

1. Access the MultiChannel Console.
2. Open the Customer Chat page.
3. Open the Queue ID drop-down list box and select the queue that you are logged on to.

   The other fields are automatically populated with values that generate a sample chat session. You can change the values as long as valid required values are included.

   The User URL is a required field that contains the application page URL to send to the agent chat browser.

   The optional Wizard URL field represents a grab URL; the actual page that appears is determined by InitChat() parameters.

   The text box displays error messages and additional information for using the sample page.
4. Click Customer Chat.

   The customer chat window appears.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

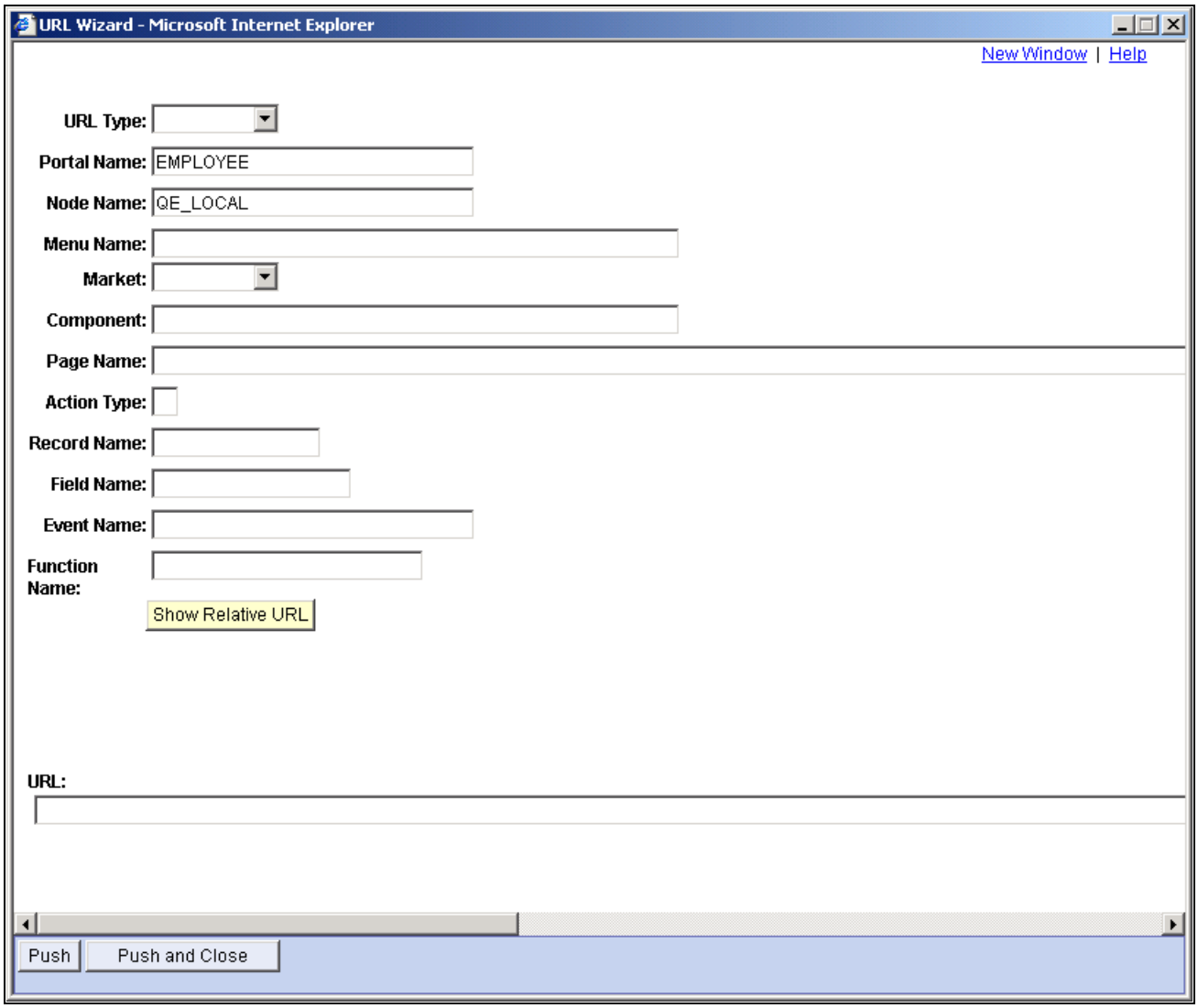
5. On the MultiChannel Console, click the flashing chat notification icon to accept the chat session.

   The agent chat window appears.

   You can enter text as both agent and customer to demonstrate chat functionality.

## Using the URL Wizard

Click Select from the agent chat window during a chat session to open the URL wizard:




A URL Wizard page having the following editable fields — URL Type, Portal Name, Node Name, Menu Name, Market, Component, Page Name, Action Type, Record Name, Field Name, Event Name, Function Name and URL

Use the wizard to form a URL to send to the customer. The URL Wizard page is a template that demonstrates constructing a URL and sending it to a customer. The URL wizard can be used as is or as the basis for developing URL generation for your application.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**URL Type** Select the type of URL that is being accessed. Values are:

- *Component:* Generates a PeopleSoft Pure Internet Architecture component URL.

  Component URL generation uses the PeopleCode built-in function **GenerateComponentContentRelURL.**

  See *PeopleTools 8.51 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions."

- *External:* Not currently used.
- *Gen URL:* Not currently used.
- *iScript:* Generates a PeopleSoft Pure Internet Architecture iScript URL.

  iScript URL generation uses the PeopleCode built-in function **GenerateScriptContentRelURL.**

  See *PeopleTools 8.51 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions."

Fields that are not required by the selected URL type are not available.

**Show Relative URL** Click to display the relative URL that is generated by the values that you have entered into the page's fields.

**URL** Displays the generated relative URL.

**Push** Click to send the generated relative URL to the customer's chat window.

The Push button demonstrates functionality that must be included in an application-specific URL wizard.

**Push and Close** Click to send the generated relative URL to the customer's chat window and close the URL wizard.

The Push and Close button demonstrates functionality that must be included in an application-specific URL wizard.

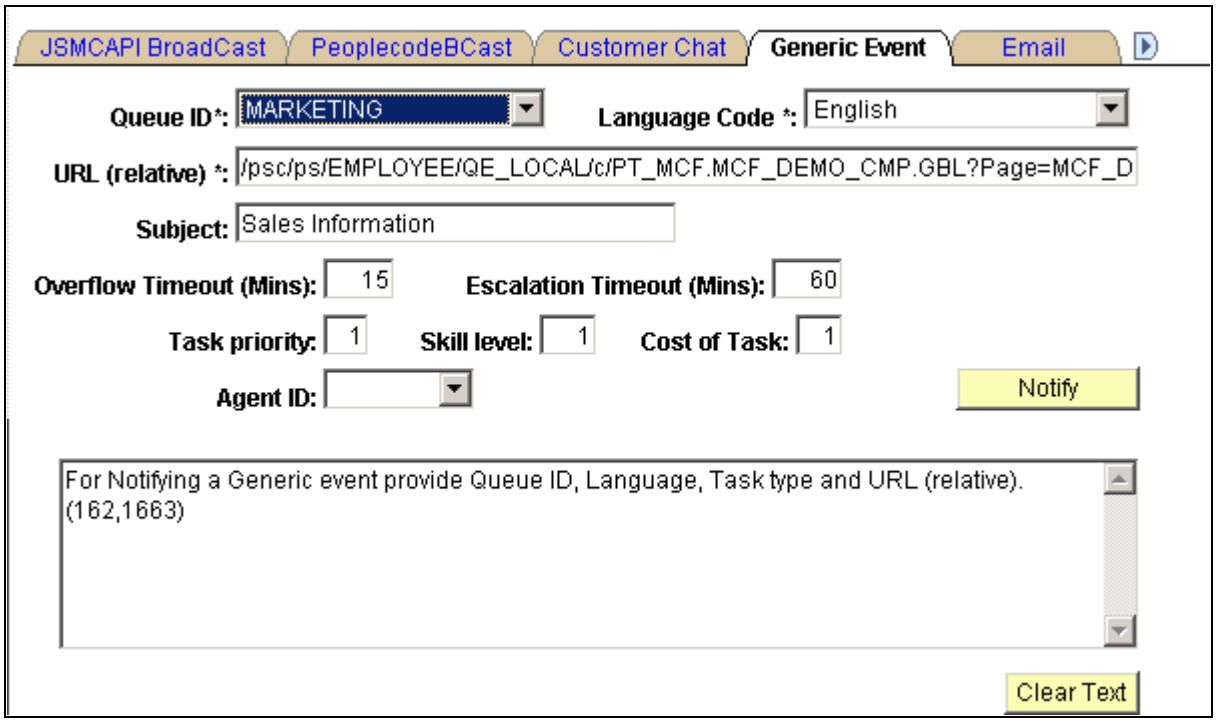
## Using the Generic Event Sample Page

Access the Generic Event page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Generic Event

**Note.** This page is available using both queue server and third-party routing server.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Generic Event page having the following editable fields — Queue ID, Language Code, URL (relative), Subject, Overflow Timeout in minutes, Escalation Timeout in minutes, Task Priority, Skill Level, Cost of Task and Agent ID

This page demonstrates a generic persistent event using the EnQueue() and NotifyQ() built-in functions, including required and optional parameters.

This page is not intended for production use. On a typical application page, the developer includes an Enqueue or similar button that calls the EnQueue() built-in function and passes required parameters, including the context of the request. The sample customer chat page demonstrates values that can be included in the EnQueue() parameters. The user may not be prompted for any information.

To run a sample generic event:

1. Access the MultiChannel Console.
2. Open the Generic Event page.
3. Open the Queue ID drop-down list box, and select the queue that you are logged on to.
4. Open the Agent ID drop-down list box, and select the agent ID that you are logged on with.
5. Select *Generic* for the task type.
6. Click Notify.
7. On the MultiChannel Console, click the flashing event notification icon to accept the event.

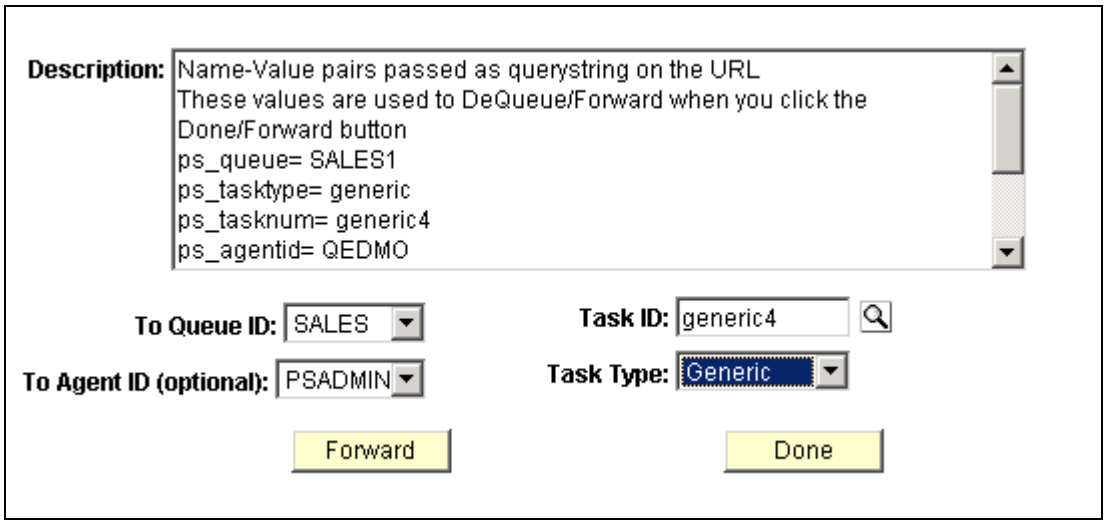
   The Generic Event window appears.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Using the Generic Event Window

The format of the generic event window is determined by application developers. PeopleSoft supplies a sample generic event window to demonstrate the available functionality, including the DeQueue() built-in.

Configure the size and initial location of the generic event window on the Agent Window Configuration page:




A Generic Event window having the following editable fields — Description, To Queue ID, Task ID, To Agent ID (Optional) and the Task Type

| | |
|---|---|
| **Description** | Displays text from the generic event. |
| **To Queue ID** | Select a queue to which the generic event will be forwarded. |
| **Task ID** | Select the task ID of the generic event to forward. |
| **To Agent ID** | Select the agent ID to whom the generic event will be forwarded. |
| **Task Type** | Select the task type of the generic event to be forwarded. |
| **Forward** | Click to send the generic event to the selected agent or queue.<br>This button demonstrates the functionality of the Forward() built-in function. |
| **Done** | Click to notify the real-time event notification (REN) server that you are done with this task. A message asks if you want to close the window.<br>The Done button demonstrates the functionality of the DeQueue() built-in function. |

### *See Also*

*PeopleTools 8.51 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions"

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Using the Email Sample Page

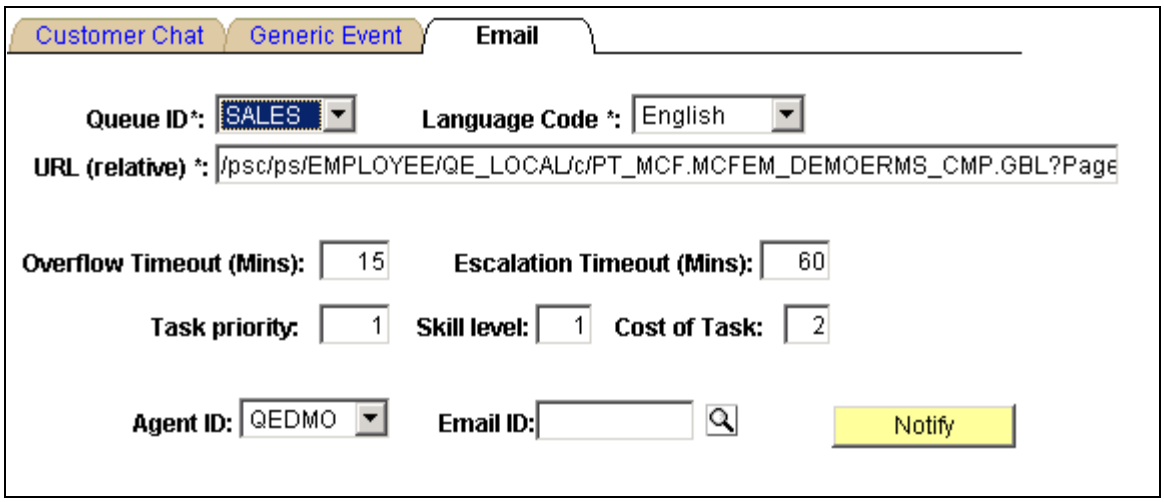
Access the Email page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Email

---

**Note.** This page is available using both queue server and third-party routing server.

---




The Email Sample page having the following editable fields — Queue ID, Language Code, URL (relative), Overflow Timeout in minutes, Escalation Timeout in minutes, Task Priority, Skill Level, Cost of Task, Agent ID, and Email ID

The email sample pages are intended for demonstration purposes and should not be used in production.

EnQueue() and NotifyQ() can also be called from PeopleSoft Application Engine batch programs.

See *PeopleTools 8.51 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions."

---

**Note.** For you to fully demonstrate the functionality of the Email sample page, an email must be read and written to the email database using the GetMail - Server sample page.

---

To process a sample email:

1. Open the Queue ID drop-down list box and select the queue that you are logged on to.
2. Open the Agent ID drop-down list box and select the agent ID that you are logged on with.
3. Click Notify.
4. On the MultiChannel Console, click the flashing email notification icon to accept the email.

   The email window appears.

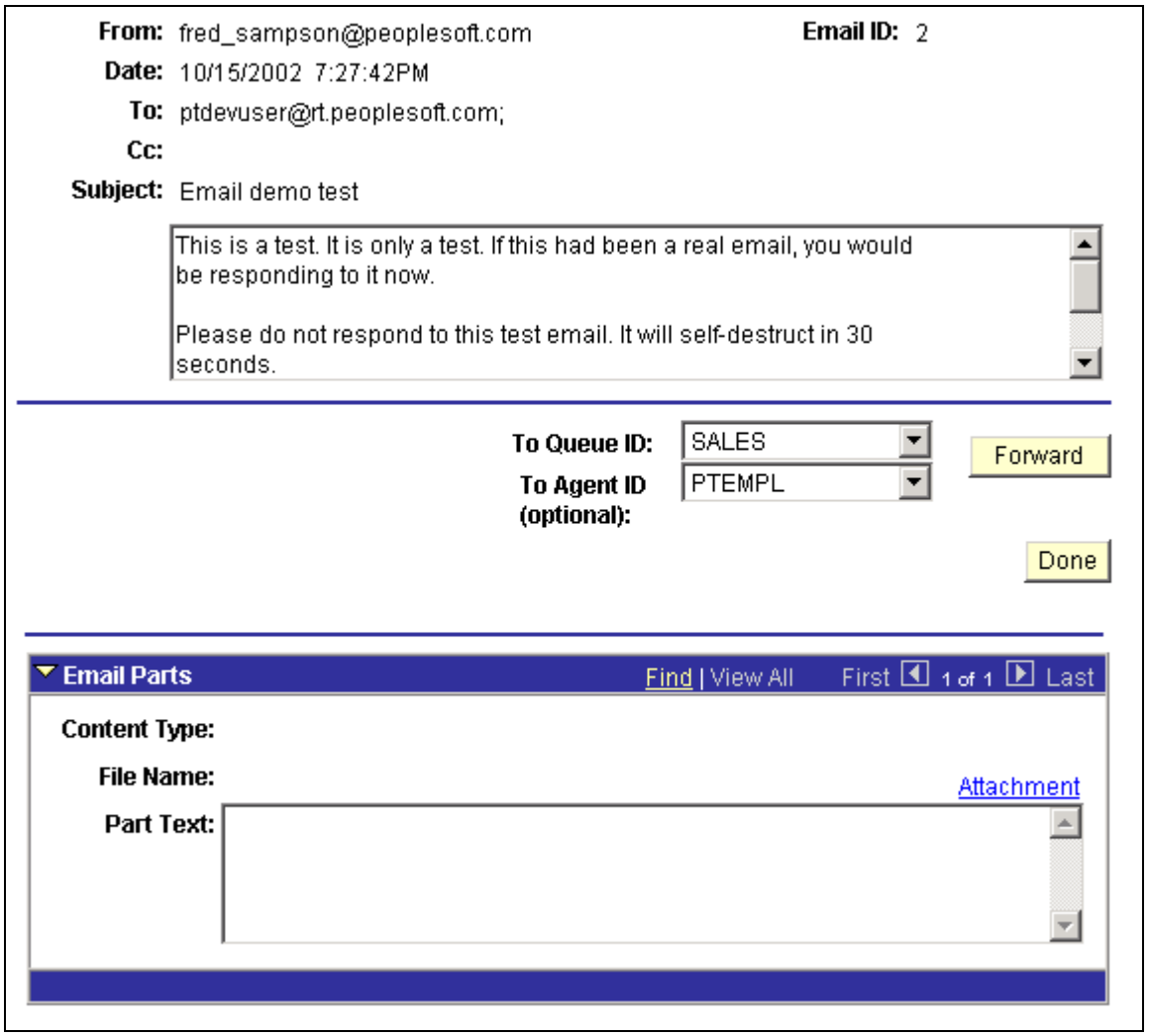

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Using the Email Window

The format of the email window is determined by application developers. PeopleSoft supplies a sample email event window to demonstrate available functionality.

Configure the size and initial location of the email event window on the agent's Window Configuration page.

The following example shows an email window:




The Email window displaying the From, Email ID, Date, To, CC and Subject and having the following editable fields — Email Text, To Queue ID and To Agent ID. This page also displays the email parts like Content Type, Attachment File Name and Part Text.

**To Queue ID** To forward the email to another queue, select a queue ID from the drop-down list box and click Forward.

**To Agent ID** To forward the email to another agent, select an agent ID from the drop-down list box and click Forward.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| | |
|---|---|
| **Forward** | Click to forward the email to the specified queue or agent.<br>This button demonstrates the functionality of the Forward() built-in function. |
| **Done** | Click to quit the email and remove it from the queue.<br>The Done button demonstrates the functionality of the DeQueue() built-in. |

#### *Email Parts*

If an email has been divided into parts stored in the email database, or has an attachment, they can be accessed here.

#### *See Also*

Chapter 14, "Configuring the Email Channel," Demonstrating the Email Channel, page 516

*PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Mail Classes"

*PeopleTools 8.51 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions"

# Using and Demonstrating JSMCAPI

To demonstrate MCF consoles and tools that use JSMCAPI, use the CTI Sample Pages (PT_CTI_DEMOOUTB) and MCF Sample Pages (MCF_DEMO_CMP) components.

This section provides an overview of JSMCAPI and discusses how to:

- Use the CTI Sample Console page and Sample CTI Console.
- Use the Agent Console page and Sample UQ Agent Console.
- Use the Monitor Agents page and Sample Monitor - Agent States page.
- Use the Monitor Queues page Sample Monitor - Queue Statistics page.

## Understanding JSMCAPI

The JSMCAPI enables custom configuration of MCF consoles (including the CTI console), MCF functionality on a PeopleSoft Pure Internet Architecture page, and queue and agent monitoring. For example, developers can create supervisor dashboards with which to monitor activity on their channels of interest, or developers can modify consoles according to their company's business requirements. The JSMCAPI builds on the REN JavaScript client. JSMCAPI uses standard JavaScript.

PeopleSoft provides sample pages demonstrating the functionality that is enabled by JSMCAPI. These pages are for demonstration purposes only, and should not be used in a production environment.

JSMCAPI is delivered with PeopleTools; you do not need a separate installation. The jsmcapi.js file is located in the <PIA_HOME>\webserv\<domain>\applications\peoplesoft\PORTAL\ps\pMCF folder.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Understanding Reason Objects and Reason Codes*

PeopleSoft MCF provides reason codes to explain any error or an event that has happened. The third party vendors uses the same reason codes as provided by PeopleSoft MCF to create reason objects that they send with the events. PSMCAPI internally parses the event, gets the reason code, and sends it to JSMCAPI with the event. The CTI sample console or third-party multichannel console looks up a mapper table that maintains the mapping of the reason codes to the Message Catalog entries and displays the reason code messages associated with the error or event that was sent by PSMCAPI.

To display the reason code messages, the sample console picks up all the reason codes and reason messages from the Message Catalogue. Whenever an error occurs, the JavaScript function will extract the reasoncode from the reason object in event and look for the reason message. Then, the JavaScript code prepares the reason message from the fields of reason object and displays the reason message in a new small window.

Reason codes can also be send with requests from JSMCAPI side. The JavaScript function will prompt for the reason codes and any extra data as part of reason.

The following table lists the reason codes and their corresponding reason messages:

| *Reason Code* | *Description/Message Entry* |
|---|---|
| 0 | System or general error has occurred. |
| 1 | See provider for details. |
| 2 | Extension or agent is busy. |
| 3 | No answer. |
| 4 | Task is in transfer. |
| 5 | Task is in conference. |
| 6 | Task is abandoned. |
| 7 | Searching for an agent. |
| 8 | Incoming task. |
| 9 | Task is assigned. |
| 10 | Connection is established. |
| 11 | User or Task data is updated. |
| 12 | Request is completed successfully. |
| 13 | State of the agent, group, or extension changed. |
| 14 | New task is initiated. |
| 15 | Conference is not successful. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Reason Code* | *Description/Message Entry* |
|---|---|
| 16 | Transfer is not successful. |
| 17 | Retrieve is not successful. |
| 18 | Forward is not successful. |
| 19 | Reject is not successful. |
| 20 | Revoke is not Successful. |
| 21 | Task is revoked. |
| 22 | Task is withdrawn. |
| 23 | Do Not Disturb is turned on. |
| 24 | Do Not Disturb is turned off. |
| 25 | Forwarding of calls is set. |
| 26 | Forwarding of calls is canceled. |
| 27 | Phone is on hook. |
| 28 | Phone is off hook. |
| 29 | New call is initiated. |
| 30 | Call is put on hold. |
| 31 | Call is parked. |
| 32 | Call is cleared. |
| 33 | Call is alternated between hold and active states. |
| 34 | Call is taken off hold. |
| 35 | Hold is not successful. |
| 36 | Invalid session. |
| 37 | Invalid State. |
| 38 | Incorrect Data. |
| 39 | Unsupported Feature. |
| 40 | Configuration Error. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Reason Code* | *Description/Message Entry* |
|---|---|
| 41 | Server is on. |
| 42 | Server is off. |
| 43 | Agent is Busy. |
| 44 | Unexpected Error. |

The following table maps the reason codes to the entries in the Message Catalog:

| *Message Number MCFMSGNUM* | *Message Set Number MCFMSGSETNUM* | *Reason Code Number REASONCODENO* |
|---|---|---|
| 2790 | 162 | 0 |
| 2791 | 162 | 1 |
| 2792 | 162 | 2 |
| 2793 | 162 | 3 |
| 2794 | 162 | 4 |
| 2795 | 162 | 5 |
| 2796 | 162 | 6 |
| 2797 | 162 | 7 |
| 2798 | 162 | 8 |
| 2799 | 162 | 9 |
| 2800 | 162 | 10 |
| 2801 | 162 | 11 |
| 2802 | 162 | 12 |
| 2803 | 162 | 13 |
| 2804 | 162 | 14 |
| 2805 | 162 | 15 |
| 2806 | 162 | 16 |
| 2807 | 162 | 17 |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| ***Message Number MCFMSGNUM*** | ***Message Set Number MCFMSGSETNUM*** | ***Reason Code Number REASONCODENO*** |
|---|---|---|
| 2808 | 162 | 18 |
| 2809 | 162 | 19 |
| 2810 | 162 | 20 |
| 2811 | 162 | 21 |
| 2812 | 162 | 22 |
| 2813 | 162 | 23 |
| 2814 | 162 | 24 |
| 2815 | 162 | 25 |
| 2816 | 162 | 26 |
| 2817 | 162 | 27 |
| 2818 | 162 | 28 |
| 2819 | 162 | 29 |
| 2820 | 162 | 30 |
| 2821 | 162 | 31 |
| 2822 | 162 | 32 |
| 2823 | 162 | 33 |
| 2824 | 162 | 34 |
| 2825 | 162 | 35 |
| 2826 | 162 | 36 |
| 2827 | 162 | 37 |
| 2828 | 162 | 38 |
| 2829 | 162 | 39 |
| 2830 | 162 | 40 |
| 2831 | 162 | 41 |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Message Number MCFMSGNUM* | *Message Set Number MCFMSGSETNUM* | *Reason Code Number REASONCODENO* |
|---|---|---|
| 2832 | 162 | 42 |
| 2833 | 162 | 43 |
| 2834 | 162 | 44 |

### *See Also*

Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Pages Used to Use and Demonstrate JSMCAPI, page 184

Chapter 13, "Understanding JSMCAPI Classes," page 251

Appendix A, "JSMCAPI Quick Reference," page 533

## Common Elements Used in This Section

This section discusses common elements used on the CTI sample pages.

### *Common Elements Used in CTI Sample Pages*

The following common elements are used in CTI sample pages.

**Agent Id** Select the CTI or MCF agent's agent ID.

In the case of a CTI agent, the agent ID may be different from the agent's user ID.

**g/e/c** (generic, email, chat) Represents three task types: generic, email, chat.

Click to initiate the action that is selected in the associated drop-down list box.

**MCF Cluster ID** Select the MCF cluster to be monitored or on which to test the sample console.

**Physical Queue** Select the physical queue to be monitored.

Each MCF logical queue can comprise one or more physical queues. Because each physical queue is associated with an MCF cluster, only physical queues can be monitored.

**State** Displays a value indicating the state of the associated element.

**Statistics** Display statistics that are generated by the CTI server.

**User Id** Displays the CTI agent's PeopleSoft user ID.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

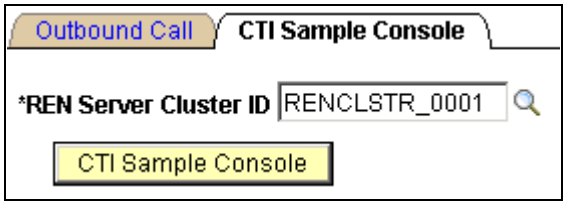
## Pages Used to Use and Demonstrate JSMCAPI

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Agent Console | PT_SAMPLEAGNTCONSL | PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Agent Console | View a sample MultiChannel Console that is generated using JSMCAPI. |
| CTI Sample Console | CTI_SAMPLECONSOLE | PeopleTools, MultiChannel Framework, CTI Configuration, Sample Pages, CTI Sample Console | View a sample CTI console that is generated using JSMCAPI. |
| Monitor Agents | PT_UQSAMPLECNSL | PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Monitor Agents | View a sample agent statistics monitoring page. |
| Monitor Queues | PT_UQSAMPLECNSQ | PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Monitor Queues | View a sample queue statistics monitoring page. |

## Using the CTI Sample Console

Access the CTI Sample Console page using the following navigation path:

PeopleTools, MultiChannel Framework, CTI Configuration, Sample Pages, CTI Sample Console




The CTI Sample Console page having the editable field REN Server Cluster ID and the CTI Sample Console button.

**REN Server Cluster ID** Select the REN server cluster on which to test the sample console.

**CTI Sample Console** Click to initiate the sample console.

**Note.** To demonstrate the CTI sample console, you must have a CTI server running and communicating with the specified REN server cluster.

After clicking CTI Sample Console, a new browser window appears that displays the sample console along with a tracer window. The Sample CTI Console contains the following group boxes which are explained in detail in the following sections:


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Open/Close Session
- Register User to Session
- Register Group, Login User to Group, and User-Group States
- Register Extension to Session, and Extension Operations
- Buddies
- Error Message/Information
- Server State and Broadcasts

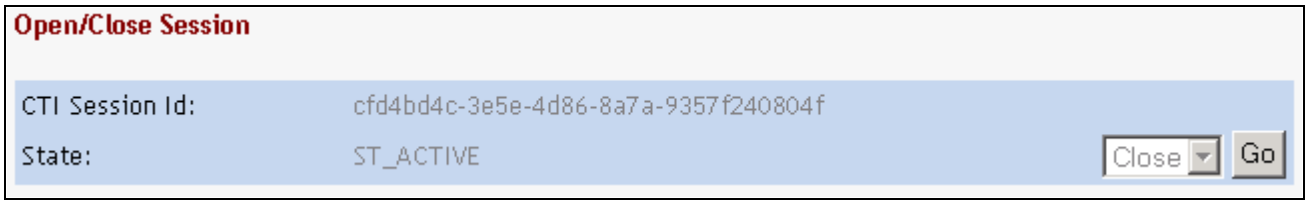
---

**Note.** To enable the new browser window to appear, disable any pop-up blocking software for your browser.

---

### *Open/Close Session*

Access the Open/Close Session group box.




Open/Close Session group box

The Open/Close Session group box displays information about the session.

**CTI Session Id** This field displays a value identifying the session.

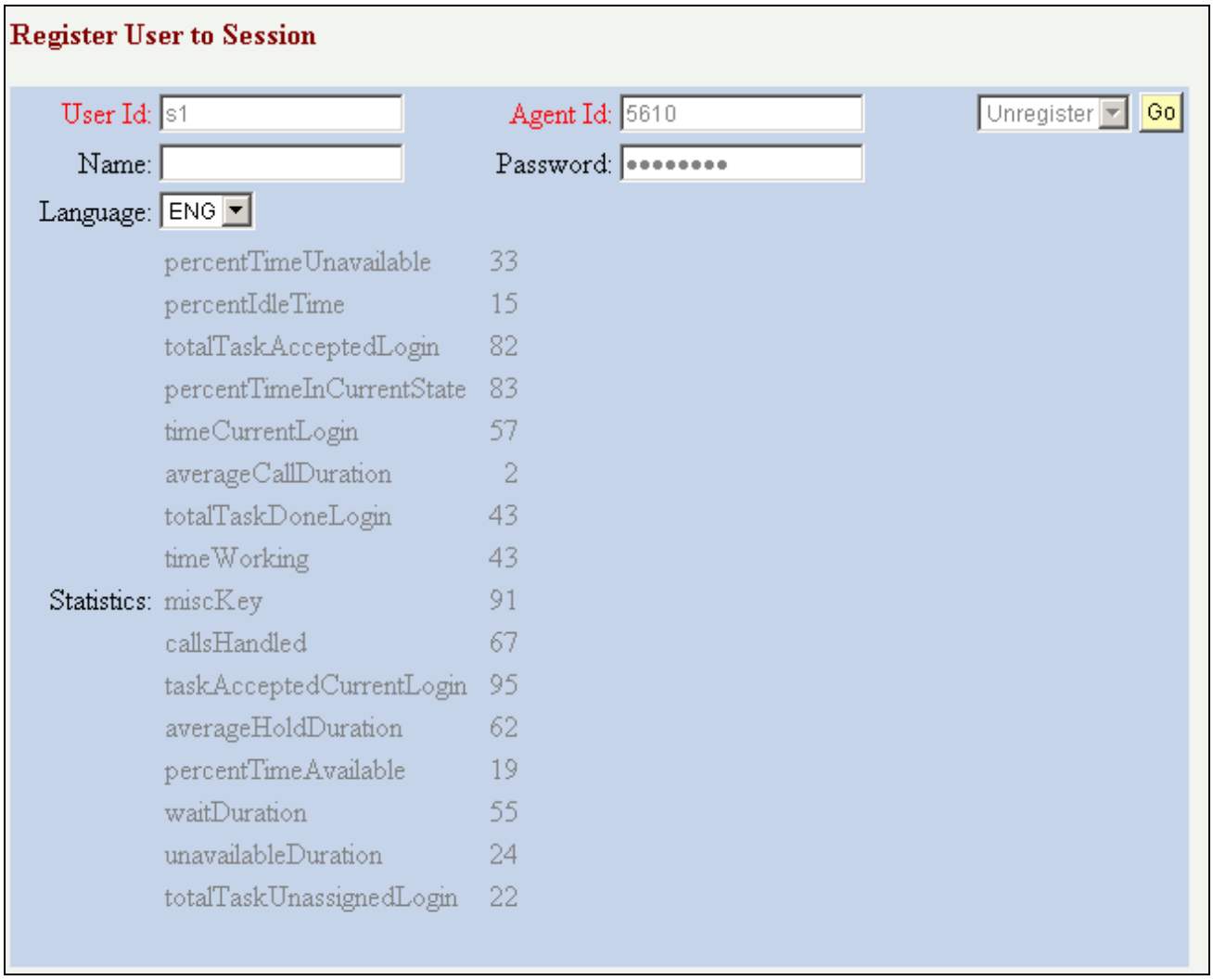
If no session ID is displayed, check to ensure that the CTI server and REN server are both running.

You can close an active session by clicking the Go button. You cannot make a selection from the drop-down list box.

### *Register User to Session*

Access the Register User to Session group box.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Register User to Session group box

Register a user to the session in the Register User to Session group box.

Because a CTI agent can have an agent ID different from the agent's user ID, both values are required. If the agent has a password, the password is also required.

**Language** Reserved for future use.

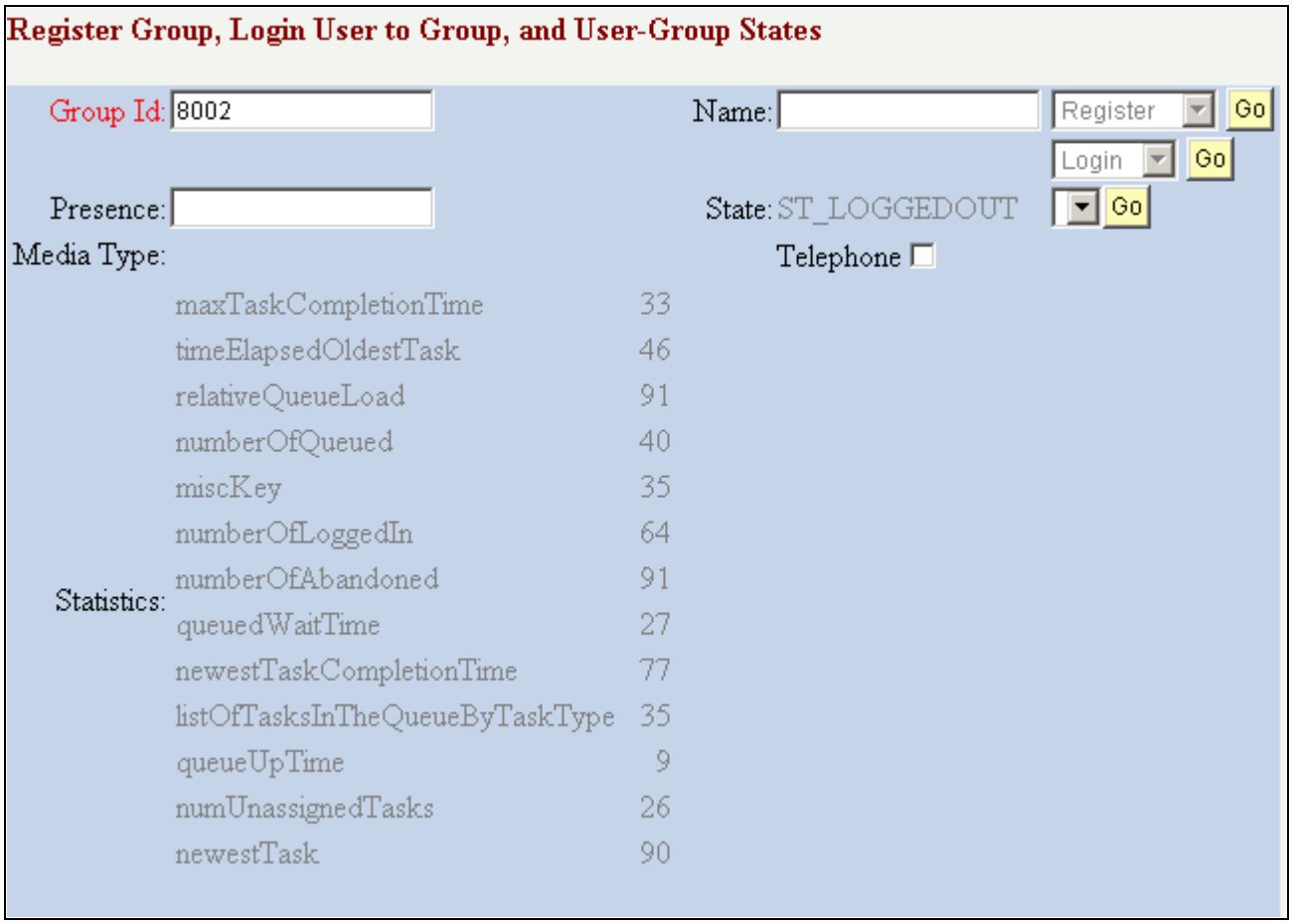
The default language is English.

You can register or unregister an agent by clicking the Go button. You cannot make a selection from the drop-down list box.

### *Register Group, Login User to Group, and User-Group States*

Access the Register Group, Login User to Group, and User-Group States group box.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Register Group, Login User to Group, and User Group States group box

**Group Id**

Enter the group ID to which the agent will be registered.

To register the agent with the specified group, click the Go button. You cannot make a selection from the drop-down list box.

**Login**

Click the Go button to log in if *Login* appears in the drop-down list box.

Click the Go button to log out if *Logout* appears in the drop-down list box.

---

**Note.** The agent cannot receive calls until the agent is in the Ready state.

---

**Note.** After you login to the group, a new pop-up window opens prompting for reason code data. If you register to an invalid group, the system displays an error message.

---

See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Understanding JSMCAPI, page 178.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Presence** Enter a presence description to associate with the state that is selected in the drop-down list box.

This presence value is different from any presence values that are predefined in the agent configuration. The agent cannot select from the pre-defined values.

**State** Select an agent state.

State options are:

- *Ready*
- *Not Ready*
- *Work Ready*
- *Work Not Ready*

Each state value has an associated state code that appears in the State field.

To enable incoming calls, select *Ready* from the drop-down list box and click the Go button.

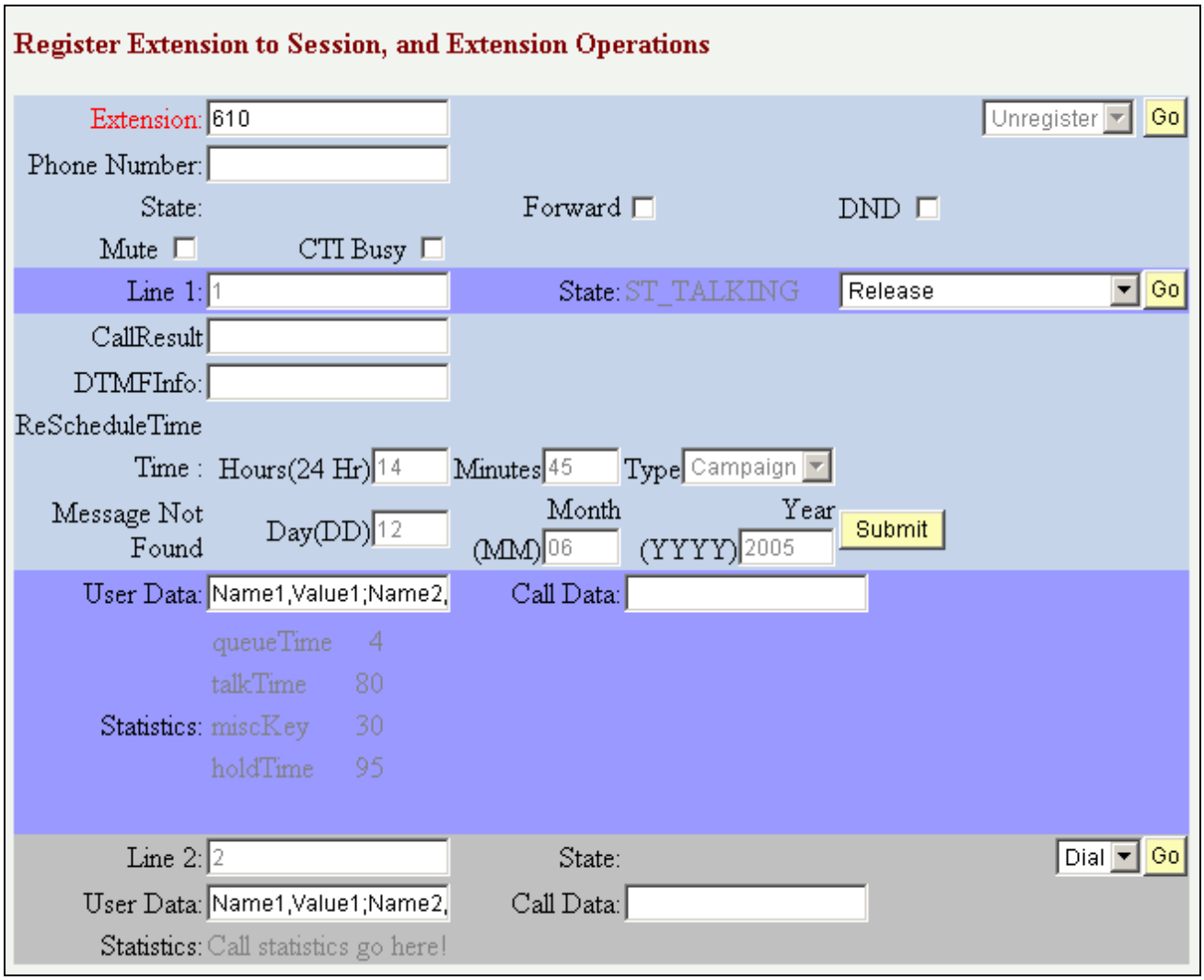
**Media Type** Reserved for future use.

The sample console supports only the telephone (voice) channel. However, JSMCAPI can support other channels.

### *Register Extension to Session and Extension Operations*

Access the Register Extension to Session, and Extension Operations group box.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Register Extension to Session and Extension Operation group box

**Extension** Enter a valid telephone extension number.

**Phone Number** Enter a valid telephone number for use in subsequent operations, such as dial, forward, or do-not-disturb (DND).

**Mute** Select to mute a call.

**Note.** This is enabled only when the agent is on a call.

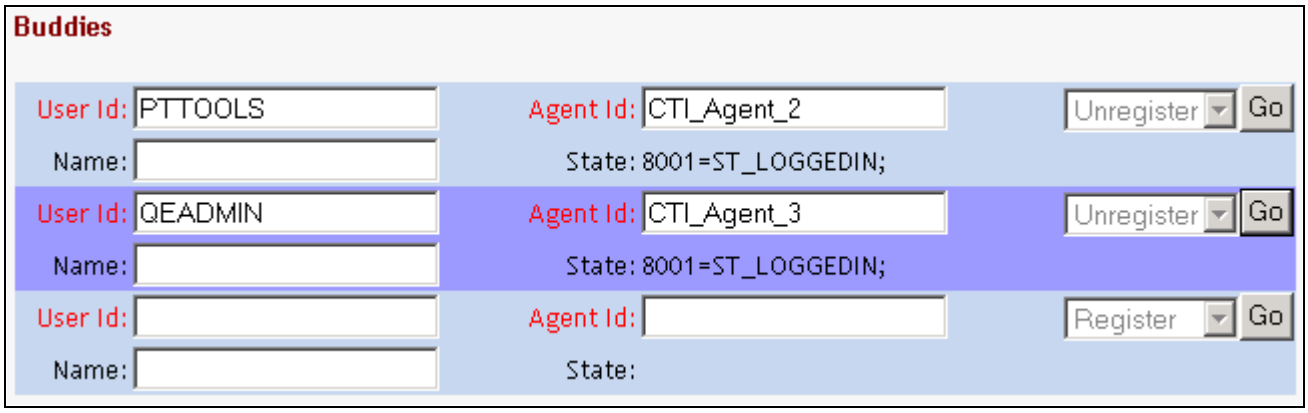
Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| | |
|---|---|
| **CTI Busy** | This flag is checked by the queue server for an agent handling a voice task.<br>**Note.** If the flag is checked, the queue server will not assign new tasks to such an agent. The CTI Busy flag allows the application to implement a request which conveys the status of the agent to the queue server. Because the queue server stores the task list and the status in its temporary cache, a recovered or a rebooted queue server will not have the information of a agent's previous CTI status. The agent can send its CTI status to the queue server to indicate that it is busy with a CTI task. |
| **Forward** | Select to forward incoming calls to the specified number. |
| **DND**(do not disturb) | Select to put the selected extension in do-not-disturb status. |
| **Line 1** or **Line 2** | Displays a name or value for each line.<br>On the sample console, the value will either be *1* or *2*. |
| **DTMFInfo** ( Dual-tone Multi frequency information) | Select to send DTMF digits on the line |
| **CallResult** | Select to set call result on the line |
| **ReScheduleTime** | Select to set reschedule time on the line |
| **Type** | Select to set reschedule type on the line. |
| **User Data** | Enter name-value pairs representing user data to be attached to a call. |
| **Call Data** | Enter any call data to be attached to the call.<br>Call data includes:<br>• *ANI*<br>• *DNIS*<br>• *THISDN*<br>• *OTHERDN* |

### *Buddies*

Access the Buddies group box.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Buddies group box

Before an agent can chat with a buddy, the buddy must be registered in the Buddies group box.

Because a CTI agent can have an agent ID that is different from the agent's user ID, both values are required.

---

**Note.** If one agent is logged on the multichannel console and another agent is logged on the sample pages, the agents may see an incorrect buddy state for each other.

---

### *Error Messages/Information*

Access the Error Messages/Information group box.

Error Messages/Information group box.

Any error messages associated with the process display in this section.

**Clear** Click the button to clear a displayed error message.

### *Server State and Broadcasts*

Access the Server State and Broadcasts group box.

Server State and Broadcasts group box

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| | |
|---|---|
| **CTI State** | Displays the status of the CTI server. |
| **REN State** | Displays the status of the REN server. |
| **Info** (information) | Displays the text of a broadcast message from the CTI server. |

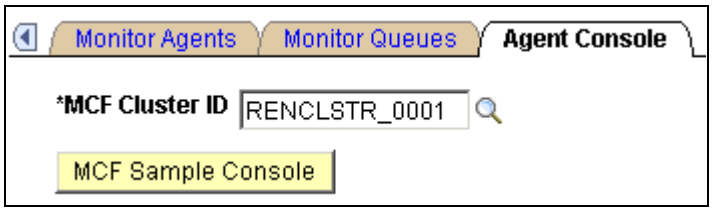
### *Tracer*

The tracer displays all requests between the CTI server and the console, which can be useful for debugging.

## Using the Agent Console Page

Access the Agent Console page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Agent Console tab




The Agent Console page having the MCF Cluster ID editable field and the MCF Sample Console button.

| | |
|---|---|
| **MCF Sample Console** | Click to initiate the sample console. |

**Note.** To demonstrate the MCF sample console, you must have an MCF cluster running and communicating with the specified REN server cluster.

After you click MCF Sample Console, a new browser window displaying the sample console appears along with a Tracer Window. The MCF Sample Page contains the following group boxes which are explained in detail in the following sections:

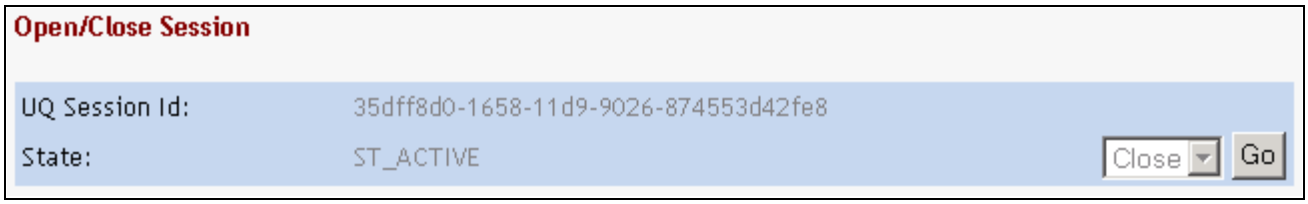
- Open/Close Session
- Register User to Session
- Register Group, Login User to Group, and User-Group States
- Address Operations
- Buddies
- Error Message/Information
- Server State and Broadcasts

**Note.** To enable the new browser window to appear, disable any pop-up blocking software for your browser.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Open/Close Session*

Access the Open/Close Session group box.

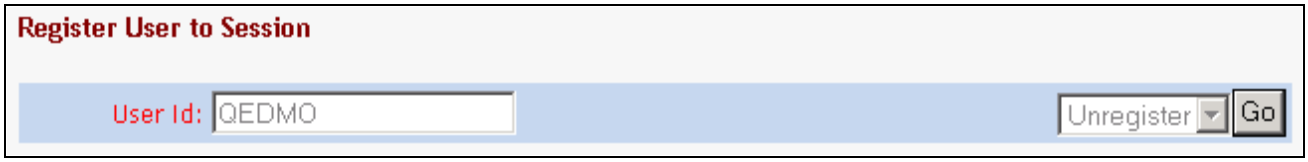



Open/Close Session group box

**UQ Session Id** The universal queue session ID.

### *Register User to Session*

Access the Register User to Session group box.




Register User to Session group box

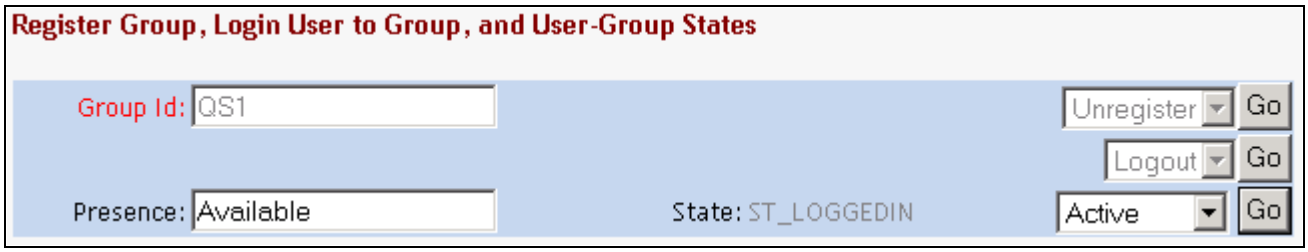
### *Register Group, Login User to Group, and User-Group States*

Access the Register Group, Login User to Group, and User-Group States group box.

---

**Note.** The presence change happens only when you change from active to inactive or from inactive to active. When you log in, the default value is active and therefore no presence change occurs.

---

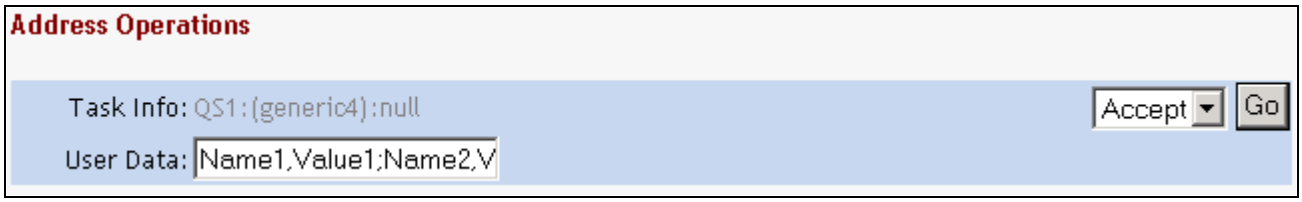



Register Group, Login User to Group, and User-Group States group box

### *Address Operations*

Access the Address Operations group box.

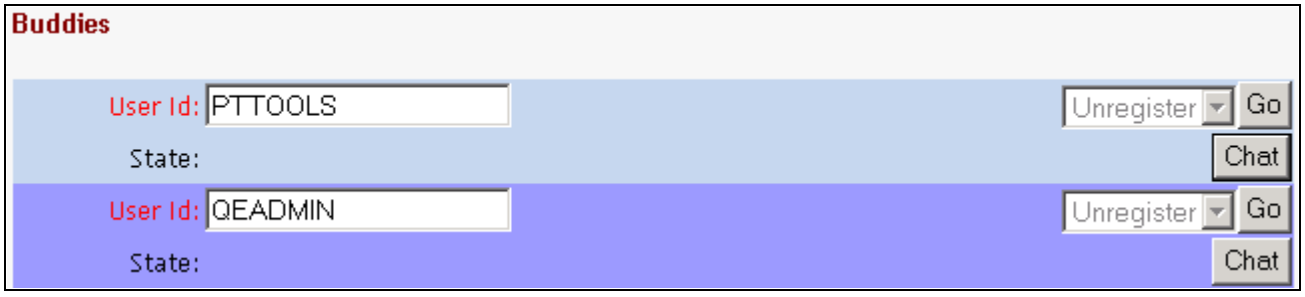
Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Address Operations group box

### *Buddies*

Access the Buddies group box.




Buddies group box

---

**Note.** If one agent is logged on the multichannel console and another agent is logged on the sample pages, the agents may see an incorrect buddy state for each other.

---

---

**Note.** The buddy states on this page are not automatically displayed. You must monitor an agent first using the Monitor Agents page before the buddy states are displayed.

---

See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the Monitor Agents Page and Sample Monitor - Agent States Page, page 195.

Click to initiate an agent-to-agent chat session.

### *Error Messages/Information*

Access the Error Messages/Information group box.

Error Messages/Information group box

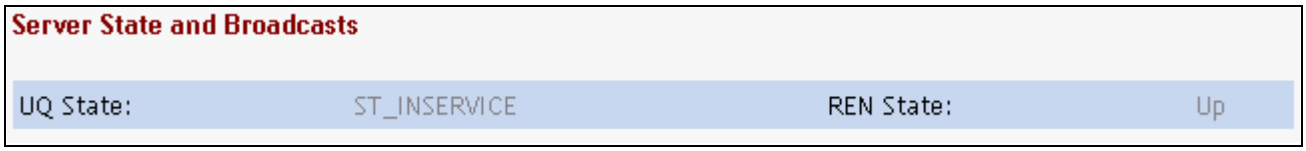
Any error messages associated with the process display here.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Clear** Click the button to clear a displayed error message.

### *Server State and Broadcasts*

Access the Server State and Broadcasts group box.




Server State and Broadcasts group box

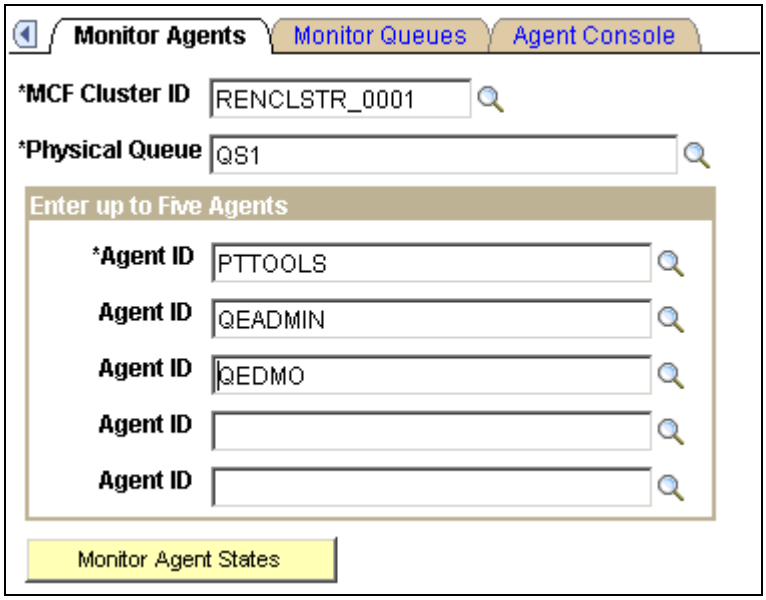
**UQ State** Displays the state of the queue server.

**REN State** Displays the state of the REN server.

## Using the Monitor Agents Page and Sample Monitor - Agent States Page

Access the Monitor Agents page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Monitor Agents tab



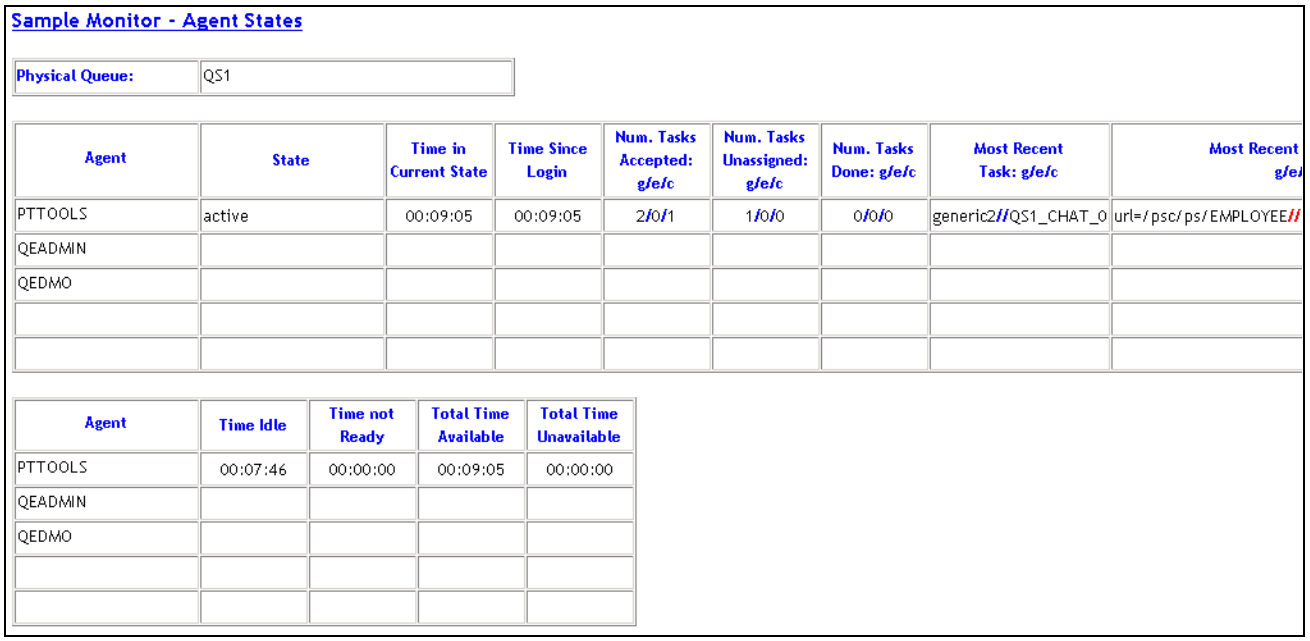

The Monitor Agents page having the MCF Cluster ID, Physical Queue, and the Agent ID editable fields and the Monitor Agent States button.

Select the MCF cluster, physical queue on that cluster, and up to five agents on that queue to be monitored.

After you click Monitor Agent States, a new browser window appears.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

**Note.** To enable the new browser window to appear, disable any pop-up blocking software for your browser.



Sample Monitor - Agent States

Physical Queue: QS1

| Agent | State | Time in Current State | Time Since Login | Num. Tasks Accepted: g/e/c | Num. Tasks Unassigned: g/e/c | Num. Tasks Done: g/e/c | Most Recent Task: g/e/c | Most Recent g/e/ |
|---|---|---|---|---|---|---|---|---|
| PTTOOLS | active | 00:09:05 | 00:09:05 | 2/0/1 | 1/0/0 | 0/0/0 | generic2//QS1_CHAT_0 | url=/psc/ps/EMPLOYEE// |
| QEADMIN | | | | | | | | |
| QEDMO | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

| Agent | Time Idle | Time not Ready | Total Time Available | Total Time Unavailable |
|---|---|---|---|---|
| PTTOOLS | 00:07:46 | 00:00:00 | 00:09:05 | 00:00:00 |
| QEADMIN | | | | |
| QEDMO | | | | |
| | | | | |
| | | | | |

Sample Monitor - Agent States page

### *Physical Queue*

The ID of the physical queue that is being monitored appears in the first table.

### *Agent Statistics Table 1*

The first agent statistics table displays information that is published by the JSMCAPI user.onStat1 event.

**Time in Current State** Displays the duration of the agent's current state.

The value returned is `timeInCurrentState`.

**Time Since Login** Denotes the duration since the agent logged in.

The value returned is `timeLogin`.

**Num. Tasks Accepted: g/e/c**(number of tasks accepted) Displays the number of tasks in each task type that the agent has accepted since login.

The value returned is `numTaskAccepted`.

**Num. Tasks Unassigned: g/e/c**(number of tasks unassigned) Displays the number of tasks in each task type that have been unassigned from the agent.

Returned by `numTasksUnassigned`.

**Num. Tasks Done: g/e/c** (number of tasks done) Displays the number of tasks in each task type that the agent has completed.

Returned by `numTasksDone`.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Most Recent Task: g/e/c** Identifies the most recent task that was accepted since login for each task type.

Returned by `mostRecentTaskId`.

**Most Recent Task Data: g/e/c** Displays data on the most recent task that was accepted since login for each task type.

Returned by `mostRecentTaskInfo`

### *Agent Statistics Table 2*

The second agent statistics table displays information that is published by the JSMCAPI user.onStat2 event.

**Time Idle** Displays the duration of time during which the agent has been idle.

An agent is considered to be idle from the time the agent logs on until the agent accepts a task, and during the time between completing the last accepted task and accepting another task.

The value returned is `timeIdle`.

**Time not Ready** Displays the duration of time during which the queue could not assign tasks to the agent because the agent's maximum workload has been reached.

The value returned is `timeNotReady`.

**Total Time Available** Displays the total amount of time that the agent has been in an available status.

The value returned is `totalTimeAvailable`.

**Total Time Unavailable** Displays the total amount of time that the agent has been in an unavailable status.

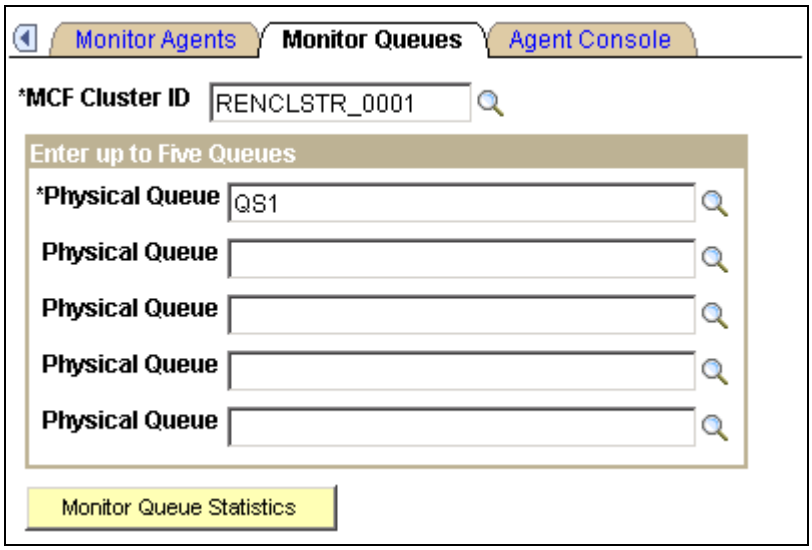
The value returned is `totalTimeUnavailable`.

## Using the Monitor Queues Page and Sample Monitor - Queue Statistics Page

Access the Monitor Queues Page using the following navigation path:

PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Monitor Queues tab

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Monitor Queues page having the MCF Cluster ID and Physical Queue editable fields, and the Monitor Queue Statistics button

Select the MCF cluster and up to five physical queues on that cluster to be monitored.

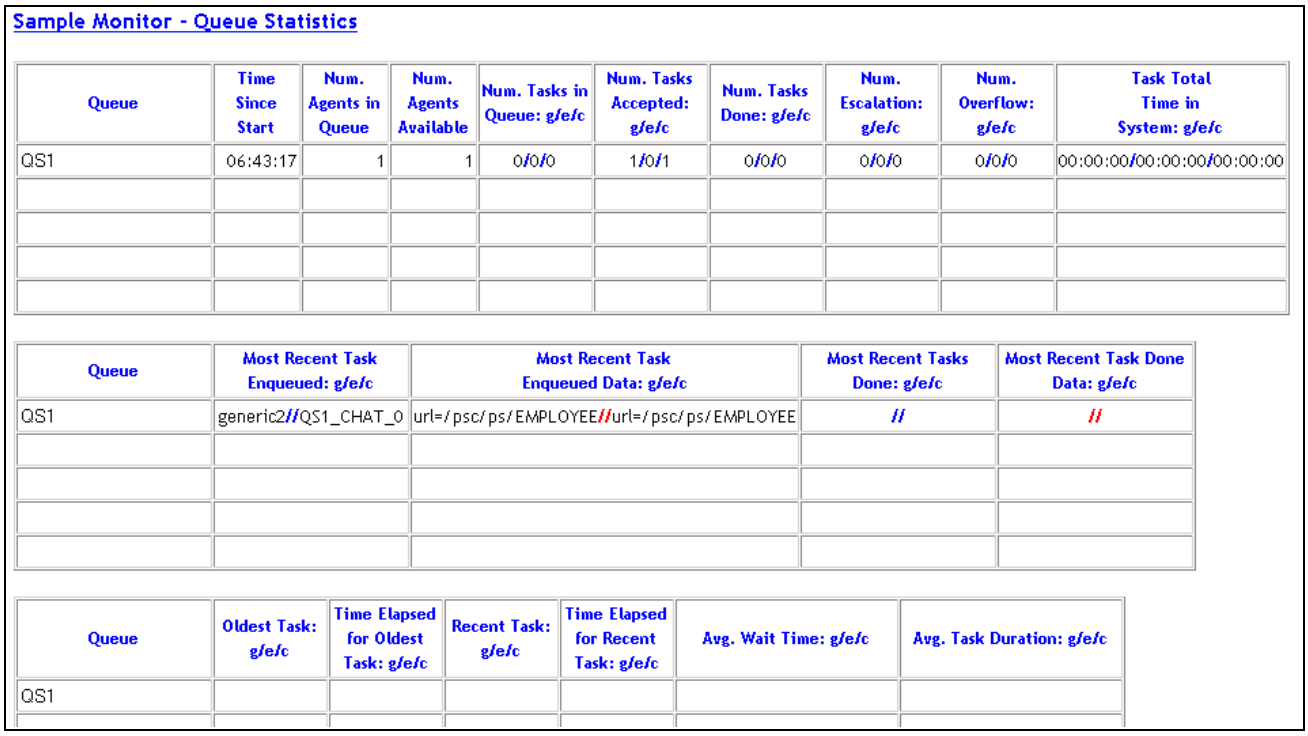
After you click Monitor Queue Statistics, a new browser window appears.

---

**Note.** To enable the new browser window to appear, disable any pop-up blocking software for your browser.

---


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.



Sample Monitor - Queue Statistics

| Queue | Time Since Start | Num. Agents in Queue | Num. Agents Available | Num. Tasks in Queue: g/e/c | Num. Tasks Accepted: g/e/c | Num. Tasks Done: g/e/c | Num. Escalation: g/e/c | Num. Overflow: g/e/c | Task Total Time in System: g/e/c |
|---|---|---|---|---|---|---|---|---|---|
| QS1 | 06:43:17 | 1 | 1 | 0/0/0 | 1/0/1 | 0/0/0 | 0/0/0 | 0/0/0 | 00:00:00/00:00:00/00:00:00 |

| Queue | Most Recent Task Enqueued: g/e/c | Most Recent Task Enqueued Data: g/e/c | Most Recent Tasks Done: g/e/c | Most Recent Task Done Data: g/e/c |
|---|---|---|---|---|
| QS1 | generic2//QS1_CHAT_0 | url=/psc/ps/EMPLOYEE//url=/psc/ps/EMPLOYEE | // | // |

| Queue | Oldest Task: g/e/c | Time Elapsed for Oldest Task: g/e/c | Recent Task: g/e/c | Time Elapsed for Recent Task: g/e/c | Avg. Wait Time: g/e/c | Avg. Task Duration: g/e/c |
|---|---|---|---|---|---|---|
| QS1 | | | | | | |

Sample Monitor - Queue Statistics page

### *Queue Statistics 1*

The first queue statistics table displays data that does not change frequently. The table displays information that is published by the JSMCAPI group.onStat1 event.

**Queue**
Displays an identifier for this queue.
The value returned is `Queue_Id`.

**Time Since Start**
The duration since this queue was started (application server domain start).
The value returned is `timeSinceStart`.

**Num. Agents in Queue** (number of agents in queue)
The number of agents that are currently logged in to this queue.
The value returned is `numAgentsLoggedIn`.

**Num. Agents Available** (number of agents available)
The number of agents that are currently logged in to this queue with an available status.
The value returned is `numAgentsAvail`.

**Num. Tasks in Queue: g/e/c**(number of tasks in queue)
The number of tasks that are currently on this queue, by task type.
The value returned is `numTasksQueued`.

**Num. Tasks Accepted: g/e/c**(number of tasks accepted)
The number of accepted tasks that are currently on this queue, by task type.
The value returned is `numTaskAccepted`.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Num. Tasks Done: g/e/c** (number of tasks done)
The number of tasks on this queue that have been completed, by task type.
The value returned is `numTaskDone`.

**Num. Escalation: g/e/c** (number of escalation)
The number of escalated tasks on this queue, by task type.
The value returned is `numEscalation`.

**Num. Overflow: g/e/c** (number of overflow)
The number of overflow tasks on this queue, by task type.
The value returned is `numOverflow`.

**Task Total Time in System: g/e/c**
Displays the total duration of the most recently completed task in the system. The total time is the difference between the time that the task was enqueued and the time at which it was marked done.
The value returned is `taskTotalTimeInSystem`.

### *Queue Statistics 2*

The second queue statistics table displays data that changes frequently. The table displays information that is published by the JSMCAPI group.onStat1 event.

**Most Recent Task Enqueued: g/e/c**
Identifies the task that was most recently enqueued on this queue, by task type.
The value returned is `mostRecentTaskEnqueued`.

**Most Recent Task Enqueued Data: g/e/c**
Displays data for the most recently enqueued task on this queue, by task type.
The value returned is `mostRecentTasksEnqueuedData`.

**Most Recent Tasks Done: g/e/c**
Identifies the task that was most recently completed, by task type.
The value returned is `mostRecentTaskDone`.

**Most Recent Task Done Data: g/e/c**
Displays data for the most recently completed tasks, by task type.
The value returned is `mostRecentTaskDoneData`.

### *Queue Statistics 3*

The table displays information that is published by the JSMCAPI group.onStat2 event.

**Oldest Task: g/e/c**
Identifies the oldest enqueued and accepted task on this queue, by task type.
The value returned is `oldestTask`.

**Time Elapsed for Oldest Task: g/e/c**
Displays the duration that the oldest task on this queue has been enqueued, by task type.
The value returned is `timeElapsedOldestTask`.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| | |
|---|---|
| **Recent Task: g/e/c** | Identifies the most recent task that was enqueued and accepted on this queue, by task type.<br>The value returned is `recentTask`. |
| **Time Elapsed for Recent Task: g/e/c** | Displays the duration that the most recent task on this queue has been enqueued, by task type.<br>The value returned is `timeElapsedRecentTask`. |
| **Avg. Wait Time: g/e/c** (average wait time) | Displays the average time between a task being enqueued and being accepted, by task type.<br>The value returned is `averageWaitTime`. |
| **Avg. Task Duration: g/e/c** (average task duration) | Displays the average duration before a task is completed.<br>The value returned is `averageTaskDuration`.<br>The average task duration is calculated by referencing a list of completed tasks. The size of that list is determined by the donelistsize parameter that is set on the Cluster Tuning page.<br>See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Tuning Cluster Parameters, page 108. |

# Using PeopleCode Built-in Functions with PeopleSoft MultiChannel Framework

PeopleSoft MultiChannel Framework uses several PeopleCode built-in functions. Application developers use these functions to communicate task requests and parameters to the queue server. For example, use InitChat code behind a button on an application page to initiate a chat session with an agent.

The built-in functions are:

- DeQueue

  Use DeQueue to notify the queue server that an enqueued task has been completed and to remove the task from the queue.

- EnQueue

  Use EnQueue to add a task to an active physical queue belonging to the specified logical queue.

- Forward

  Use Forward to transfer a task from one agent to another agent or another logical queue.

- InitChat

  Use InitChat to place a customer chat request on a queue.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

- NotifyQ

  Use NotifyQ to notify the queue server of a task enqueued by EnQueue.

See *PeopleTools 8.51 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions."

## Using Universal Queue Classes

The universal queue classes provide the means by which applications can inspect objects that are processed by the queue server, such as tasks, agents, and queues. The classes also enable tasks to reenter the queue with their original task ID, as well as keep task times based on their original entry into the system.

See *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Universal Queue Classes."

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 12

# Configuring PeopleSoft MCF for Third-Party Routing Systems

In previous releases, PeopleSoft CTI required a third-party middleware in the form of the Genesys CTI Framework, a Cisco ICM system, or CTI middleware that implements PSMCAPI.

Beginning with PeopleTools 8.48, PeopleSoft MultiChannel Framework (MCF) uses third-party multichannel routing systems to offer a wide range of communication channels to empower PeopleSoft applications like CRM. In the third-party system, the queue is the logical storage unit for representing a work item, and no difference exists between a logical and physical queue. The third party develops the universal routing system to route email, voice, chat, and generic tasks.

PeopleSoft customers may choose either the pre-8.48 features of queue server and CTI or the new third-party routing system to use email, chat, voice, and generic channels.

This chapter provides an overview of third-party routing systems and discusses how to:

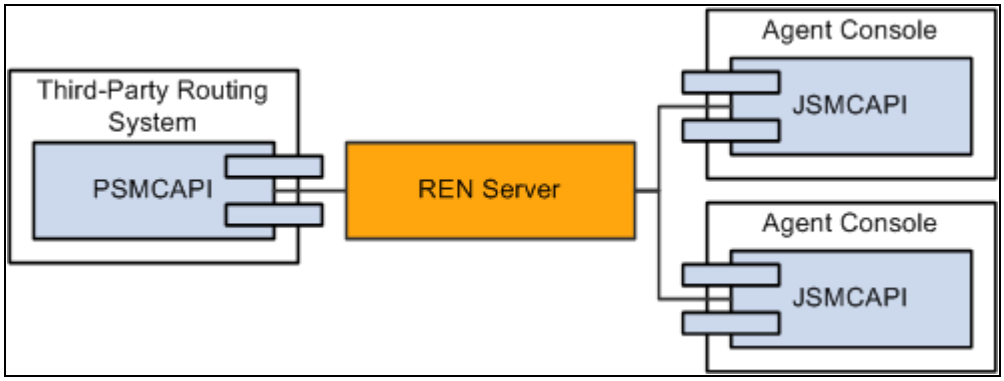
- Configure PeopleSoft MCF for a third party.
- Define the third-party flag.
- Define the PeopleSoft MCF Cluster page for a third party.
- Tune PeopleSoft MCF cluster parameters for a third party.
- Notify PeopleSoft MCF clusters of changed parameters for a third party.
- Define PeopleSoft MCF queues for a third party.
- Define PeopleSoft MCF agents for a third party.
- Configure PeopleSoft MCF tasks for a third party.
- Configure CTI for a third party.
- Communicate with customers and agents using chat.
- View event logs for a third party.
- View broadcast logs for a third party.
- Work with third-party sample pages.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Understanding Third-Party Routing Systems

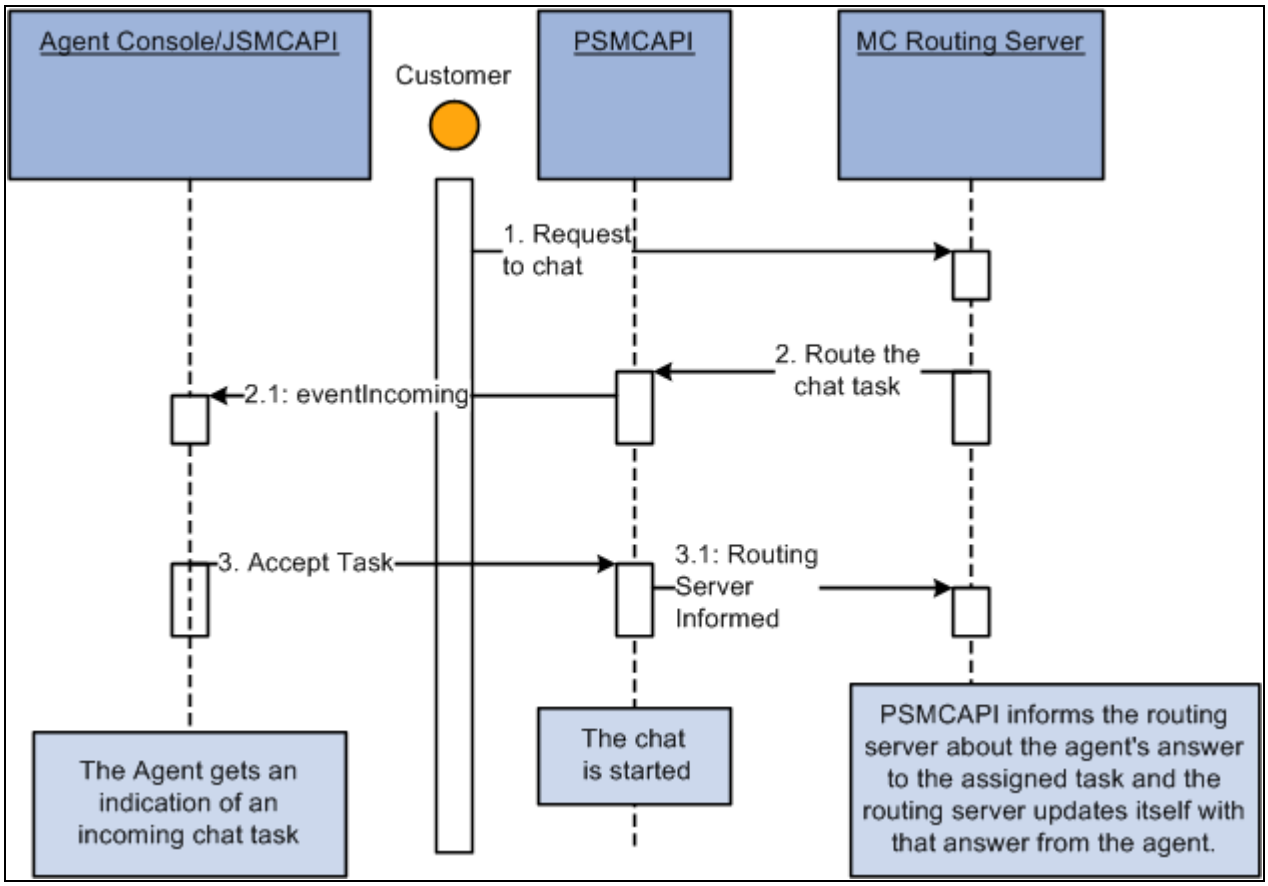
PSMCAPI provides an interface to the third-party routing system that enables communication between various PeopleSoft components and the third party. An event, like an incoming task from the third-party routing system, is received by PSMCAPI and pushed to JSMCAPI (MCF console) using the REN server. Similarly, a request from MCF console or JSMCAPI is sent to PSMCAPI using the REN server and is eventually passed on to the third-party routing system. The REN server routes requests and events internally to the PeopleSoft system, including an agent desktop or browser.

The following diagrams illustrate how PSMCAPI acts as an interface to the third-party routing system and communicates with JSMCAPI through the REN server:




PSMCAPI interface with PeopleSoft MCF


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

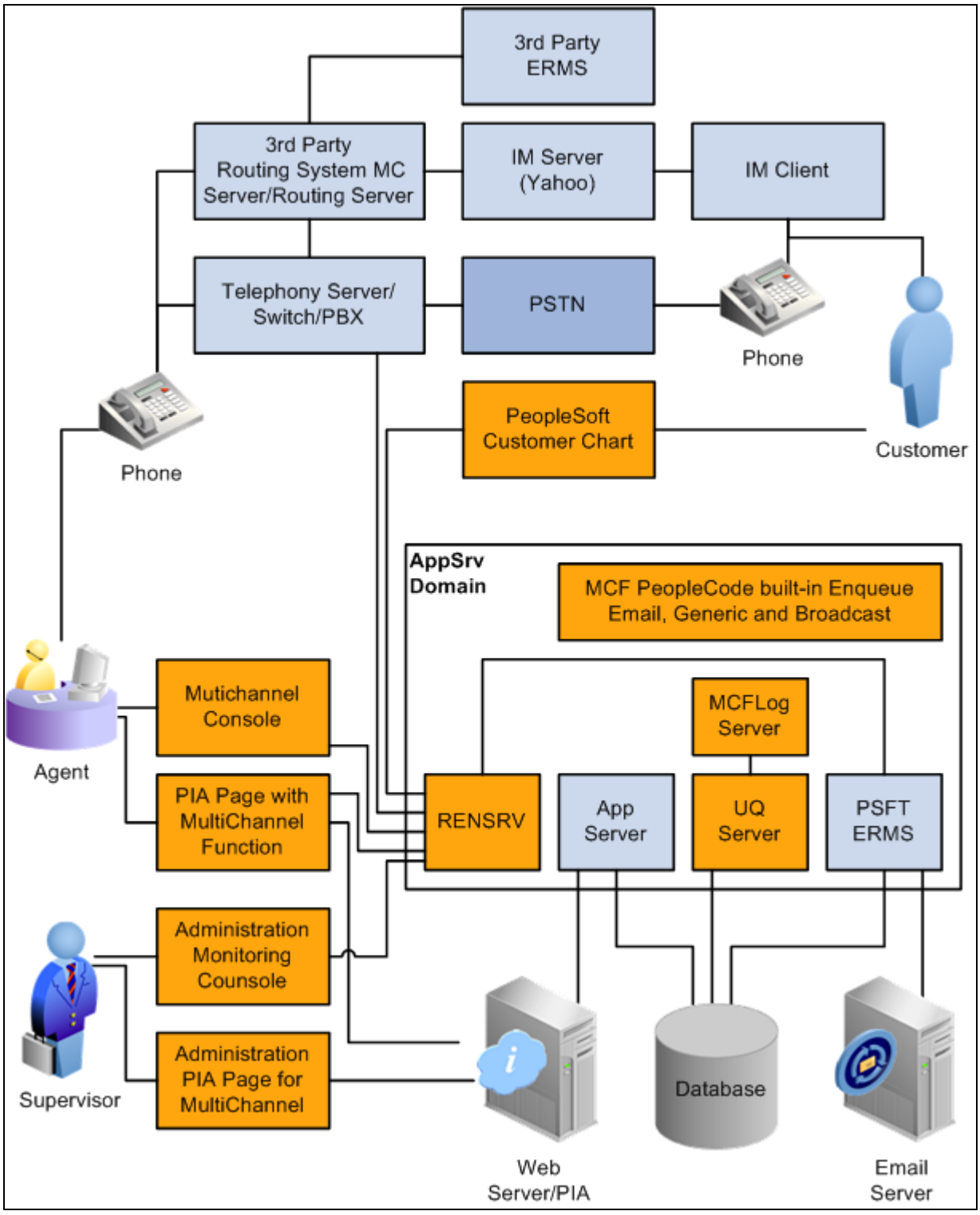



Interaction of PSNCAPI with third-party and JSMCAPI

PSMCAPI and JSMCAPI support multiple channels of communication, queue, and agent statistics information that enable the leverage of third-party routing systems.

The following diagram illustrates how the third-party routing system interacts with PeopleSoft MCF:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Third-party routing system with PeopleSoft MCF


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

The third-party routing system connects to all kinds of media servers, and routes the task to the agent based on routing rules. The third-party routing system connects to the REN server through HTTP or HTTPS using PSMCAPI. All requests sent by the agents to third-party system such as login, logout, accept incoming call, and forward email use the above connection. All media requests sent by PeopleCode built-in functions to the REN server such as InitChat or Enqueue also use the above connection. All events from the routing system such as assignment, broadcast, and statistics, go to the agent or the supervisor through this connection.

In the third-party routing system, the queue server acts as an overflow or escalation adapter that writes the overflow or escalated tasks from the third-party routing system to the PeopleSoft Database. When using a third-party routing system for escalated and overflowed task, the queue server is used for task insertions to the database. The third-party routing system processes and maintains task properties and timers. On a task time-out, the routing system sends an event to the queue server to insert the task in the database. The routing system creates the events to remove the task from the agents and recalculates their workload.

This section discusses how to:

- Define third-party routing system requirements.
- Define third-party routing rules.

## Defining Third-Party Routing System Requirements

In general, the third-party router:

- Allows business users to define routing strategies.
- Defines escalation and timeout routing paths.
- Allows agents to receive any type of work from multiple channels and queues simultaneously or in parallel.
- Allows pushed and pulled events simultaneously or in parallel.
- Defines intra- and extra-channel escalations.

## Defining Third-Party Routing Rules

Third-party routing rules are based on:

- Task type, for example, email, chat, voice, scheduled callback, fax, and so on.
- Time against SLA (Service Level Agreement) in form of overflow and escalation timers.
- Priority of the event expressed as a positive single-digit number.
- Skill within organization or task type.
- Primary language required to resolve any issue.
- Cost of each event type assigned by the administrator to avoid overloading.

The agents or supervisors must have the capabilities to:

- Change the skills, language, priority, or cost of an item to affect its subsequent routing.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Assign an item to a specific queue or agent.
- Accept, reject, forward, mark as resolved, or mark as awaiting additional information.

## Configuring PeopleSoft MCF for a Third Party

This section lists the basic steps to configure PeopleSoft MCF for a third party. Subsequent sections of this PeopleBook describe each step in detail.

You complete these steps to configure PeopleSoft MCF for a third party:

1. Define the third-party flag.

   See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Defining the Third-Party Flag, page 209.

2. Define PeopleSoft MCF clusters for a third party.

   See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Defining the PeopleSoft MCF Cluster Page for a Third Party, page 210.

3. Tune PeopleSoft MCF clusters for a third party.

   See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Tuning PeopleSoft MCF Cluster Parameters for a Third Party, page 212.

4. Notify PeopleSoft MCF of changed parameters for a third party.

   See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Notifying PeopleSoft MCF Clusters of Changed Parameters for a Third Party, page 215.

5. Define PeopleSoft MCF queues for a third party.

   See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Defining PeopleSoft MCF Queues for a Third Party, page 217.

6. Define PeopleSoft MCF agents for a third party.

   See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Defining PeopleSoft MCF Agents for a Third Party, page 220.

7. Configure PeopleSoft MCF tasks for a third party.

   See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Configuring PeopleSoft MCF Tasks for a Third Party, page 231.

8. Configure CTI.

   See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Configuring CTI for a Third Party, page 232.

9. View event logs.

   See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Viewing Event Logs for a Third Party, page 236.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

10. View broadcast logs.

    See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Viewing Broadcast Logs for a Third Party, page 238.

11. Work with third-party sample pages.

    See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Working with Third-Party Sample Pages, page 239.

# Defining the Third-Party Flag

This section discusses how to define the third-party flag.

To define the third-party flag, use the Third-party Flag (MCF_TP_FLAG_CMP) component.

## Page Used to Define the Third-Party Flag

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Third-Party Flag | MCF_TP_FLAG_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Third-Party Flag | Select this flag to use the third-party routing system to configure MCF. |

## Defining the Third-Party Flag

Access the Third Party Flag page using the following navigation path:

PeopleTools, MultiChannel Framework, Third-Party Configuration, Third-Party Flag

The Third-Party Flag page showing the Use Third Party Routing check box

The default value of the third-party flag is *False*. If the flag is set to *False*, the queue server routes chat, generic, and email and uses PeopleSoft CTI to route voice. If the flag is set to *True*, all events, chat, generic, email, and voice are routed by the third-party routing system, and you can configure MCF using third-party pages only. The queue server:

- Stops listening to all routing requests.
- Starts up new listeners for overflow and escalation.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Separate database tables are created for queue server and third-party routing server that combine all fields from MCF and CTI. When the third-party flag is set to *True*, all pages using queue server are disabled and grayed out. When the third-party flag is set to *False*, all pages using third-party routing system are disabled and grayed out.

**Note.** Whenever you change the value of this flag, you must reboot the queue server and MCF Log server.

# Defining the PeopleSoft MCF Cluster Page for a Third Party

This section discusses how to define the PeopleSoft MCF Cluster page for the third party. To view the Cluster page, use the Cluster (MCF_TP_RENCFG_CMP) component.

## Page Used to Define the PeopleSoft MCF Cluster Page for a Third Party

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Cluster | MCF_TP_UQCLUSTER | PeopleTools, MultiChannel Framework, Third-Party Configuration, Cluster | View information about third-party clusters. |

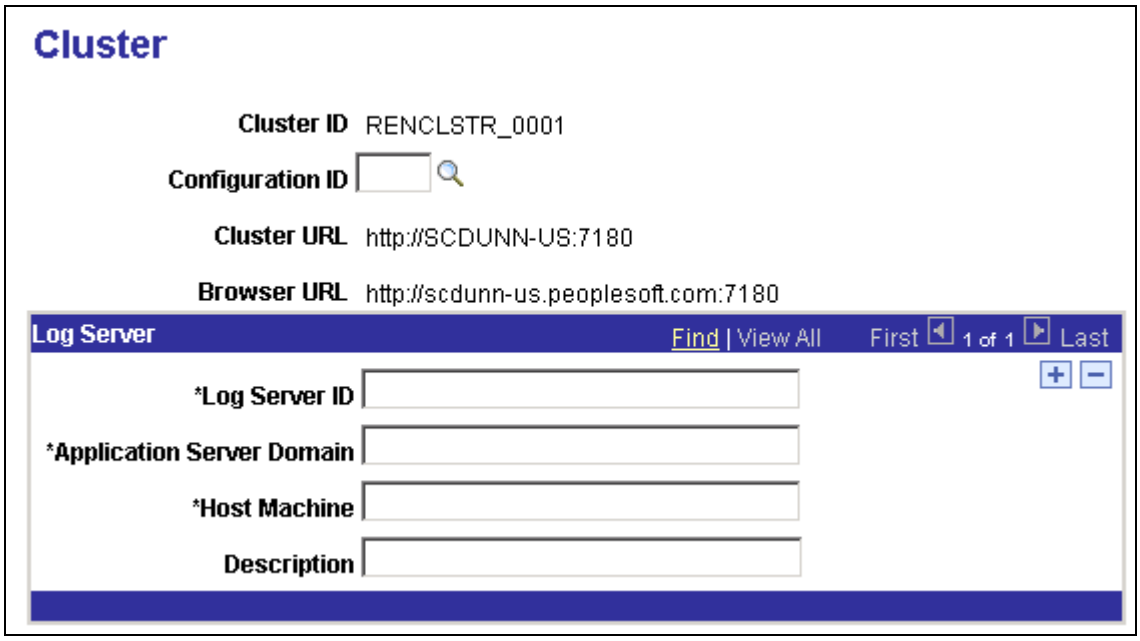
## Defining the PeopleSoft MCF Cluster Page for a Third Party

The Cluster page displays the associated PeopleSoft MCF cluster URLs and log server details for the selected cluster.

Access the Cluster page using the following navigation path:

PeopleTools, MultiChannel Framework, Third-Party Configuration, Cluster

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Cluster page displaying the Cluster ID, Cluster URL, and the Browser URL and having the following editable fields: Configuration ID, Log Server ID, Application Server Domain, Host Machine, and Description.

**Cluster ID** Displays the cluster ID of the cluster containing the queue server and log server.

**Configuration ID** Displays the configuration ID.

**Note.** This value is required only for CTI configuration. To select the configuration ID from lookup, define the configuration ID when you are configuring CTI.

See Chapter 3, "Configuring PeopleSoft Computer Telephony Integration," Configuring CTI Console Type, page 24.

**Cluster URL** Displays the URL for the REN server that serves this cluster.

**Note.** The Cluster URL is defined on the REN Server Definitions page.

See Chapter 5, "Configuring REN Servers," Defining REN Servers, page 77.

**Browser URL** Displays the browser URL.

**Note.** The browser URL may be different from the cluster URL, which should not have to go through any firewall, reverse proxy server, or other outward-facing security barrier.

The browser URL is same as defined on the REN Server Definitions page.

See Chapter 5, "Configuring REN Servers," Defining REN Servers, page 77.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Log Server*

A cluster consists of a primary log server and any number of backup servers. Each cluster requires a minimum of one log server and one queue server. The primary log server is the first log server started and the remaining log servers are backups. If the primary log server fails, the system determines the subsequent primary log server among the backups.

---

**Note.** You can save a cluster without creating a log server or a queue server. But, for MCF to run, you need at least one log server and one queue server.

---

You can add a log server to a cluster by adding a new row. Before removing a log server, ensure that it is not the master, then shut down its domain. Then click Delete (the minus sign).

**Log Server ID** Enter a unique identifier for each log server to identify its entries in the log cluster table.

The queue server process paired with this log server uses this same ID to identify its entry in the database table.

**Application Server Domain** Enter the application server domain of which this log server is a member.

**Host Machine** Enter the name of the application server host machine.

**Description** Enter the description of the host machine.

# Tuning PeopleSoft MCF Cluster Parameters for a Third Party

This section discusses how to tune PeopleSoft MCF cluster parameters for a third party. Tuning cluster parameters may give you better performance. To tune cluster parameters, use the Cluster Tuning (MCF_TP_SYS_NV_CMP) component.

## Page Used to Tune PeopleSoft MCF Cluster Parameters for a Third Party

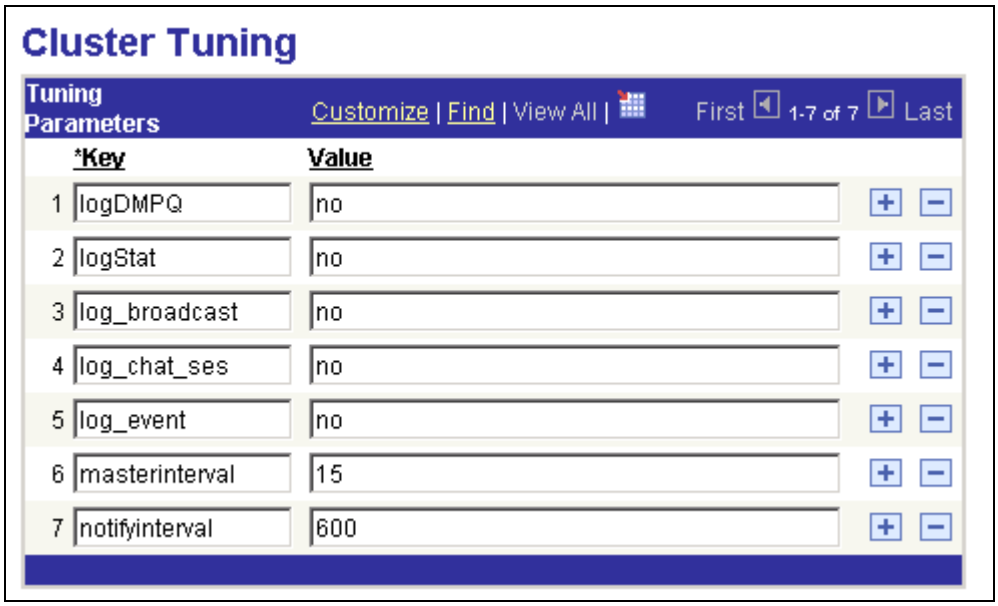
| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Cluster Tuning | MCFTP_SYSTEM_KV_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Cluster Tuning | Modify cluster parameters to tune performance. |

## Tuning PeopleSoft MCF Cluster Parameters for a Third Party

Access the Cluster Tuning page using the following navigation path:

PeopleTools, MultiChannel Framework, Third-Party Configuration, Cluster Tuning

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Cluster Tuning page having the Key and Value editable fields

Use the Cluster Tuning page to set MCF cluster parameters to optimize performance or enable logging for a cluster.

If you make changes to a cluster parameter, you must use the third-party Notify Cluster page to propagate the changes.

See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Notifying Third-Party Clusters of Changed Parameters, page 216.

The following table lists the cluster tuning parameters you can modify and describes the default values and usage of each:

| ***Key*** | ***Default value*** | ***Usage*** |
|---|---|---|
| logDMPQ | *No* | Select *Yes* if you want PSMCFLOG to log REN server event notifications resulting from bcastinterval broadcasts.<br>Only the event is logged, not its contents.<br>This is a logging parameter. If you change the value of this parameter you must use the Refresh Logging Parameters button on the third-party Cluster Notify page to notify clusters of the changed parameter<br>See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Notifying Third-Party Clusters of Changed Parameters, page 216 and Chapter 9, "Administering Queues, Logs, and Tasks," Viewing Event Logs, page 144. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Key* | *Default value* | *Usage* |
|---|---|---|
| logStat | *No* | Select *Yes* to log the statistics returned by the queue server for the onStat1 user and group events to the database.<br>This is a logging parameter. If you change the value of this parameter you must use the Refresh Logging Parameters button on the third-party Cluster Notify page to notify clusters of the changed parameter<br>See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Notifying Third-Party Clusters of Changed Parameters, page 216 and Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using and Demonstrating JSMCAPI, page 178. |
| log_broadcast | *No* | Select *Yes* to turn on logging of the broadcast messages that are sent.<br>This is a logging parameter. If you change the value of this parameter you must use the Refresh Logging Parameters button on the third-party Cluster Notify page to notify clusters of the changed parameter<br>See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Notifying Third-Party Clusters of Changed Parameters, page 216 and Chapter 9, "Administering Queues, Logs, and Tasks," Viewing Broadcast Logs, page 141. |
| log_chat_ses | *No* | Select *Yes* to turn on logging of the contents of chat sessions.<br>This is a logging parameter. If you change the value of this parameter you must use the Refresh Logging Parameters button on the third-party Cluster Notify page to notify clusters of the changed parameter<br>See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Notifying Third-Party Clusters of Changed Parameters, page 216 and Chapter 9, "Administering Queues, Logs, and Tasks," Viewing Chat Logs, page 142. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Key* | *Default value* | *Usage* |
|---|---|---|
| masterinterval | *15* | Interval, in seconds, after which a cluster master updates its timestamp in its cluster tables. Slave clusters check the timestamp to determine that the master cluster is still running.<br>A lower value enables rapid discovery of a failed master server, but increases log server overhead. A higher value reduces log server overhead, but delays discovery of a failed master server.<br>If a only one log server is configured for an MCF cluster, this value can be large.<br>The masterinterval value also acts as a heartbeat interval for the master log server connection to user consoles.<br>This is a timing parameter. If you change the value of this parameter you must use the Refresh Timing Parameters button on the third-party Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Notifying Third-Party Clusters of Changed Parameters, page 216. |
| notifyinterval | 600 | Interval, in seconds, after which the database is checked for any pending enqueued tasks. The third party will be notified if any enqueued task is pending in the database.<br>This is a timing parameter. If you change the value of this parameter you must use the Refresh Timing Parameters button on the third-party Cluster Notify page to notify clusters of the changed parameter.<br>See Chapter 12, "Configuring PeopleSoft MCF for Third-Party Routing Systems," Notifying Third-Party Clusters of Changed Parameters, page 216. |

## Notifying PeopleSoft MCF Clusters of Changed Parameters for a Third Party

This section describes how to notify PeopleSoft MCF clusters of changed parameters for a third party. To notify clusters of changed parameters, use the Cluster Notify (MCF_TP_NOTIFY_CMP) component.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Page Used to Notify Third-Party Clusters of Changed Parameters

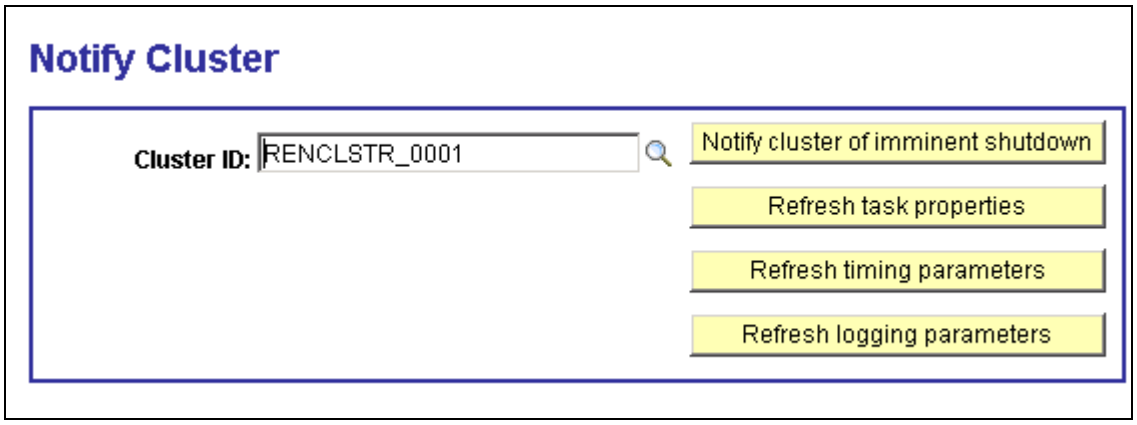
| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Notify Cluster | MCFTP_CL_NOTIFY_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Cluster Notify | Notify a cluster of changes to parameters or a configuration, or of a shutdown. |

## Notifying Third-Party Clusters of Changed Parameters

Use the Notify Cluster page to notify a cluster of certain changes to its parameters, constituent queues, or that its application servers are being shut down.

Access the Notify Cluster page using the following navigation path:

PeopleTools, MultiChannel Framework, Third-Party Configuration, Cluster Notify




The Cluster Notify page having the Cluster ID editable field and the following buttons: Notify Cluster of imminent shutdown, Refresh task properties, Refresh timing parameters, and Refresh logging parameters.

**Notify cluster of imminent shutdown** Click to send a message to every agent logged on to the selected cluster that they have been logged off.

Send this notification if the cluster's application servers are being shut down.

**Refresh task properties** Click to load task properties that have been changed on the Tasks page for the selected cluster.

**Refresh timing parameters** Click to reload parameters, other than logging parameters, changed on the Cluster Tuning page for the selected cluster.

**Refresh logging parameters** Click to reload logging parameters changed on the Cluster Tuning page for the selected cluster.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Defining PeopleSoft MCF Queues for a Third Party

To define PeopleSoft MCF queues for the third-party routing system, use the third-party queue (MCF_TP_Q_CFG_CMP) component. This section discusses how to:

- Define queues.
- Define canned queue messages.
- Define canned queue URLs.

## Pages Used to Define PeopleSoft MCF Queues for a Third Party

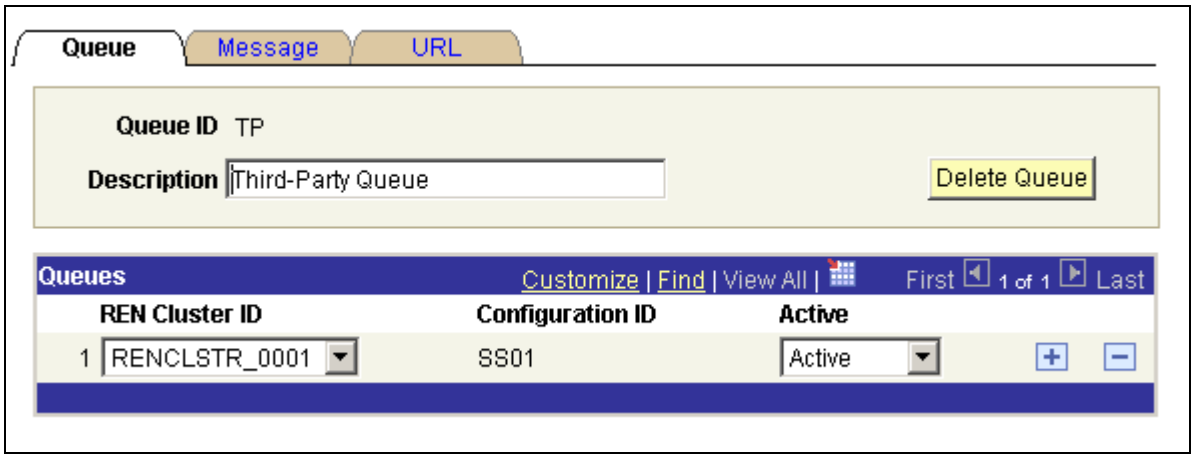
| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Queue | MCF_TP_QUEUE_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Queue | Add a queue or view the details of an existing queue. |
| Message | MCFTPSYSMSG_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Queue, Message | Define messages that every agent can use. |
| URL | MCFTPQUEUEURL_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Queue, URLs | Define URLs that each agent can push to a client. |

## Defining PeopleSoft MCF Queues for a Third Party

Access the Queue page using the following navigation path:

PeopleTools, MultiChannel Framework, Third-Party Configuration, Queue

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Queue page displaying the Queue ID and having the following editable fields: Description, REN Cluster ID, Configuration ID, and Active.

**Queue ID** Queue IDs must be alphanumeric, but they may include underscore characters.

**Configuration ID** Displays the configuration ID.

**Note.** This value is required only for CTI configuration. To select the configuration ID from lookup, define the configuration ID when you are configuring CTI.

See Chapter 3, "Configuring PeopleSoft Computer Telephony Integration," Configuring CTI Console Type, page 24.

**Delete Queue** Click to remove this queue.

Deleting a queue means that no work or agents can be assigned to the queue and the queue is removed from all agents' available queues.

**Cluster ID** The Cluster ID field identifies the cluster that services this queue.

**Active** Select *Active* or *Inactive* from the drop-down list box.

The queue server does not send new tasks to an inactive queue. Agents and existing tasks remain on an inactive queue. To receive new queued tasks, the queue must be active.

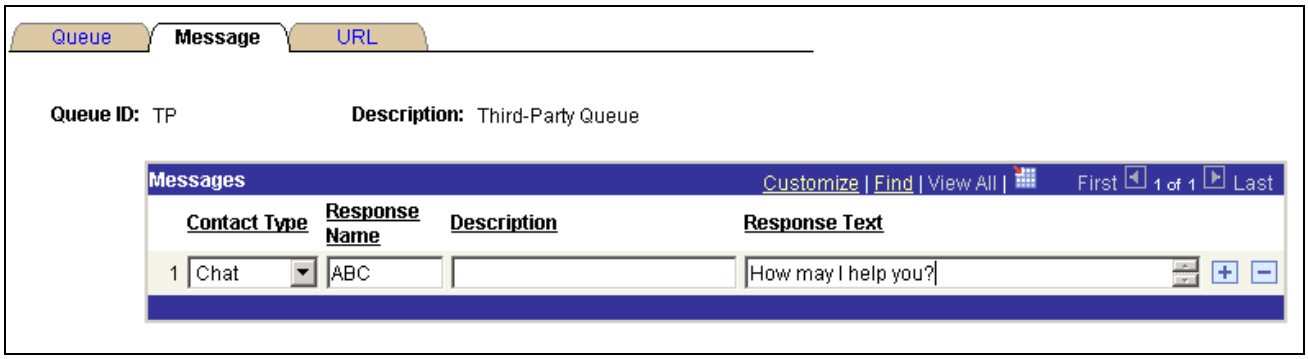
Active and inactive statuses support "follow-the-sun" practices. For example, SALES1 could be supported in the London office, and SALES2 could be supported in the San Francisco office when the London office is closed by activating and inactivating the appropriate queues.

## Defining Canned Queue Messages

Access the Message page using the following navigation path:

PeopleTools, MultiChannel Framework, Third-Party Configuration, Queue, Message


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Message page displaying the Queue ID and Description and having the following editable fields: Contact Type, Response Name, Description, and Response Text.

**Contact Type** Select one of the following contact types:

- *Chat*
- *Email*

**Response Name** The response name appears in the agent's Template Messages drop-down list box for all agents who belong to this queue.

All messages are downloaded when the agent launches the agent chat console by accepting a customer chat. Template messages are not available in collaborative chat.

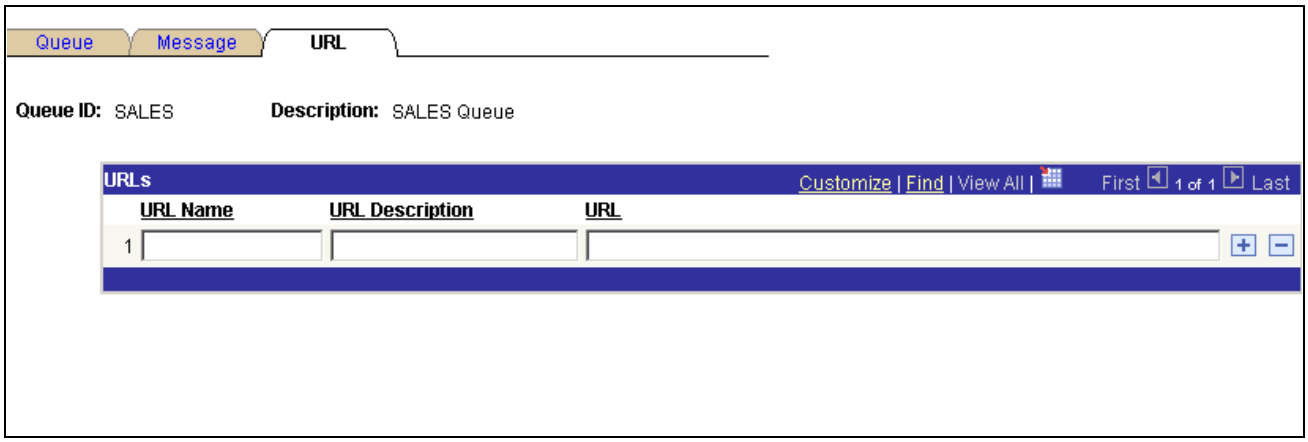
**Response Text** The specified message appears in the client chat window when selected by the agent.

## Defining Canned Queue URLs

Access the URL page using the following navigation path:

PeopleTools, MultiChannel Framework, Third-Party Configuration, Queue, URL

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The URL page displaying the Queue ID and Description and having the following editable fields: URL Name, URL Description, and URL.

Canned queue URLs enable the agent to push a predefined URL to the customer, which appears in a pop-up window for the customer. The agent can edit these URLs in the chat window after selecting them.

**URL Name** Enter the name of the URL.

The URL name appears in the agent's URL drop-down list box for all agents that belong to this queue.

**URL Description** Enter a description of the URL.

This description appears only on this page to further describe this URL or, for example, its reason for inclusion.

**URL** Enter the URL. The URL must include the protocol (http://) and any required parameters.

All canned queue URLs defined for the queue are downloaded when the agent launches the agent chat console by accepting a customer chat. These URLs are not available in collaborative chat.

If you send a canned queue URL that is a PeopleSoft Pure Internet Architecture URL, be sure that the recipient has permissions to access that portal, node, or page.

## Defining PeopleSoft MCF Agents for a Third Party

To define PeopleSoft MCF agents for a third party, use the (MCF_TP_AGENT_CMP) component. This section discusses how to:

- Create agents.
- Set up buddy lists.
- Customize windows.
- Define messages.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Specify agent-specific URLs.
- Define agent's presence.
- Specify media.
- Specify languages that an agent supports.
- Specify miscellaneous parameters.

## Pages Used to Define PeopleSoft MCF Agents for a Third Party

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Agent | MCFTP_AGENT_PG | PeopleTools, MultiChannel Framework, Third Party Configuration, Agent | Enter name, skill level, maximum workload, and queues for each agent. |
| Buddy | MCFTPAGENTBUDDY_PG | PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Buddy | Enter a list of other agents. |
| Customization | MCFTPAGENTCUST_PG | PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Customization | Configure the specified agent's window. |
| Messages | MCFTPAGENTMSG_PG | PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Messages | Enter agent-specific messages, in addition to available system messages. |
| URL | MCFTPAGENTURL_PG | PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, URL | Enter agent-specific URLs in addition to canned queue URLs. |
| Presence | MCFTPAGENTPRES_PG | PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Presence | Enter an agent's presence options. |
| Media | MCFTPAGENTMEDIA_PG | PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Media | Enter an agent's capabilities. |
| Languages | MCFTP_AGENTLANG_PG | PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Languages | Enter the languages that an agent is qualified to use. Enter at least one language. |

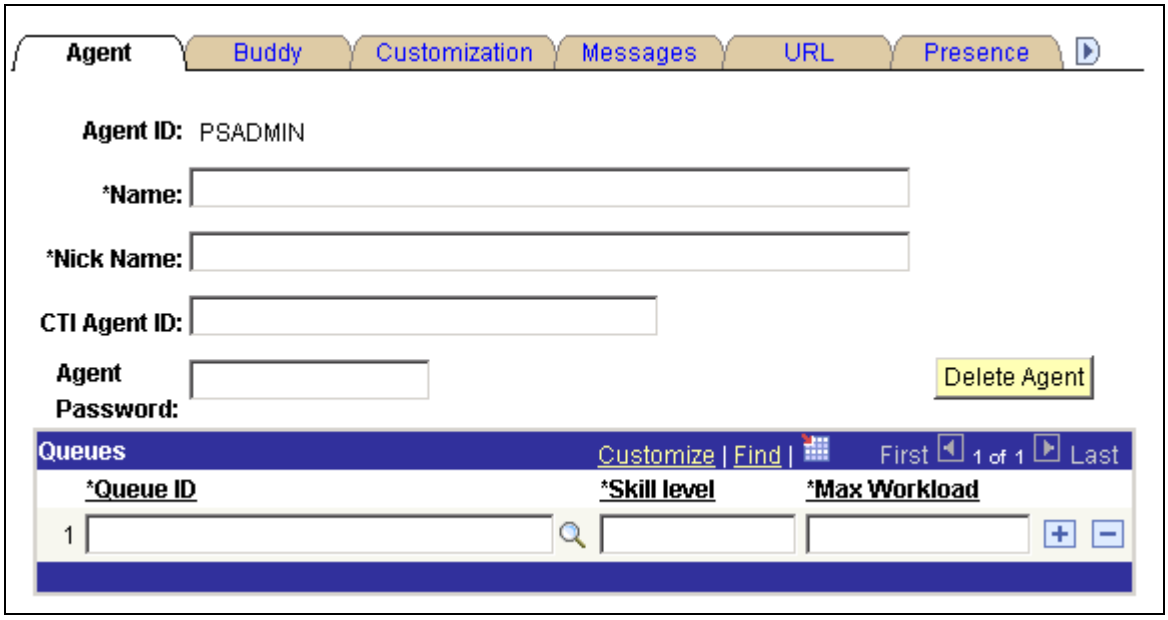
Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Page Name | Definition Name | Navigation | Usage |
|---|---|---|---|
| Miscellaneous | MCFTP_AGENTMISC | PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Miscellaneous | Enter miscellaneous behaviors. |

## Creating PeopleSoft MCF Agents for a Third Party

Access the Agent page using the following navigation path:

PeopleTools, MultiChannel Framework, Third Party Configuration, Agent




The Agent page displaying the Agent ID and having the following editable fields: Name, Nick Name, CTI Agent ID, Agent Password, Queue ID, Skill Level, and Max Workload.

**Agent ID** Specifies the agent ID.

**Name** Enter the full name of this agent in (lastname,firstname) format. The agent name appears in other agents' buddy lists.

**Nick Name** Enter a short name for this agent. The agent nickname identifies this agent in chat sessions and logs.

**CTI Agent ID** Enter a CTI agent ID for a CTI agent.

**Agent Password** Enter a password for the agent.

**Queue ID** Enter the ID of a queue to which this agent is assigned. Each agent may be assigned to more than one queue. An agent can log on to only one queue at a time from the MultiChannel Console.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Skill** Select the skill level of this agent for the tasks assigned for this queue. This field is required.

This option is used by third parties for task assignment. Third parties define skill levels based on business needs, requirements, policy, and so on.

**Maximum Workload** Select the maximum load that this agent can be assigned before tasks are held or assigned to other agents. This field is required.

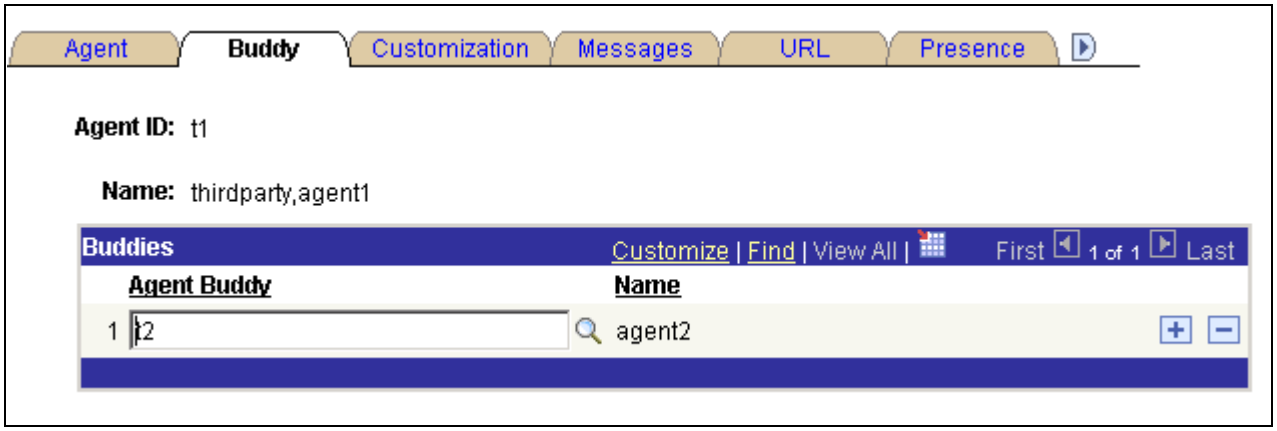
This option is used by third parties for task assignment. Third parties define maximum workload based on business needs, requirements, policy, and so on.

**Note.** Do not delete a queue from an agent's list unless that agent has no open accepted tasks on that queue.

## Setting Up Buddy Lists

Access the Buddy page using the following navigation path:

PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Buddy




The Buddy page displaying the Agent ID and Name and having the Agent Buddy and Name editable fields.

The agent's buddy list facilitates collaborative chat and chat conferencing.

**Agent Buddy** Select another agent with whom this agent can have a chat session or can ask to conference onto another chat.

**Name** Displays the buddy agent's nickname. The name appears in the buddy list on the MultiChannel Console or agent chat window.

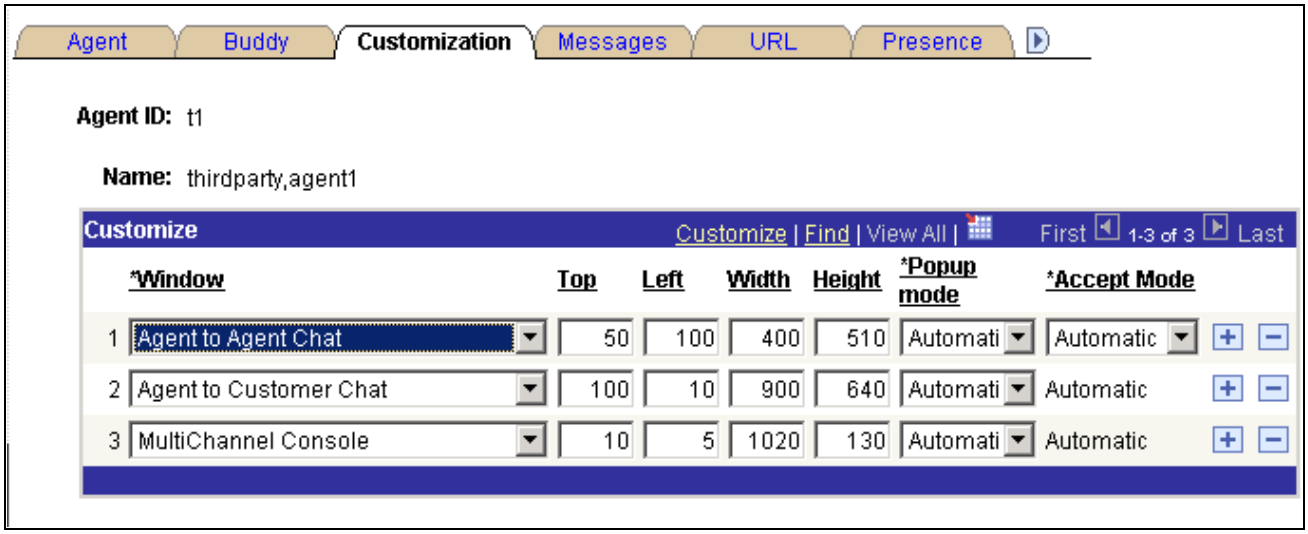
An agent's presence as shown in the buddy list on the MultiChannel Console or on the Invite Agent list on the chat console indicates the agent's availability for chat or conference.

## Customizing Windows

Access the Customization page using the following navigation path:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Customization




The Customization page displaying the Agent ID and Name and having the following editable fields: Window, Top, Left, Width, Height, Popup Mode, and Accept Mode.

Set the initial agent window placement and size by specifying parameters on this page. An agent may resize and move the windows.

**Window** Select the window to which the specified configuration applies.

Select from:

- *Agent to Agent Chat*
- *Agent to Customer Chat*
- *E-mail*
- *Generic Alert*
- *Grab URL*
- *MultiChannel Console*

**Top** and **Left** Enter the distance in pixels from the top and left edge of the screen when the window first appears.

**Width** and**Height** Enter the width and height, in pixels, of the window when it first appears.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Popup Mode**

Select from:

*Automatic:* The window appears automatically. For customer-initiated chat or tasks initiated from the EnQueue built-in function, the task is automatically accepted as well. For agent-initiated chat, the agent can elect not to accept the task; in effect, the agent can preview the task. If this is the desired behavior, select *Manual* from the Accept Mode drop-down list box. If you want agent-to-agent tasks to function like customer-initiated tasks, select *Automatic* from the Accept Mode drop-down list box.

*Manual:* The window does not appear until the agent clicks the task on the agent MultiChannel Console. For customer-initiated chat or tasks initiated from the EnQueue() built-in function, clicking the task means that the task is automatically accepted as well. For agent-initiated chat, the behavior depends on the setting for the Accept Mode field.

**Accept Mode**

Select from:

*Automatic:* Agent-to-agent chats are automatically accepted without requiring the agent to click the icon.

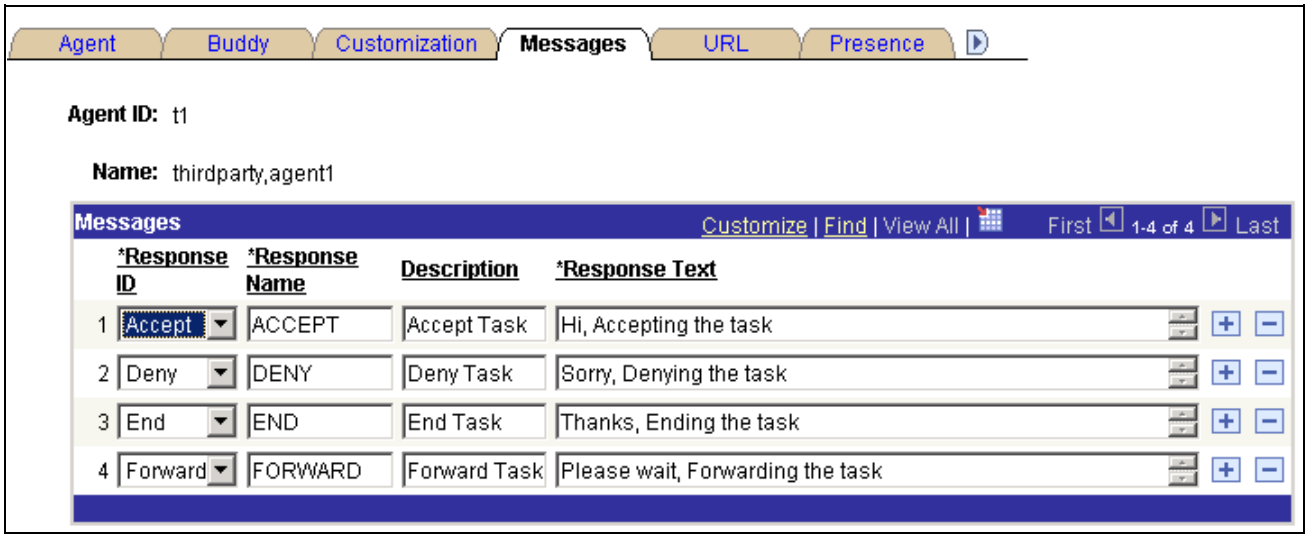
*Manual:* Agent-to-agent chats require the agent to click the icon.

Accept mode affects only collaborative chat.

## Defining Messages

Access the Messages page using the following navigation path:

PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Messages




The Messages page displaying the Agent ID and Name and having the following editable fields: Response ID, Response Name, Description, and Response Text.

An agent can create personalized responses in addition to the system responses defined for each queue.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Response ID**

Responses, except those identified as Other, are linked to specific events. These responses, except for those identified Other, are always sent on these events from this agent. If an agent does not have a customized response for a specific event, it is read from a default response set in the Message Catalog. The response text set here overrides the default text set in the Message Catalog.

Select from:

- *Abandon:* A chat is abandoned when a chat initiator closes the chat window before the chat is accepted by an agent. This response appears when the agent accepts the abandoned chat.
- *Accept:* This response is automatically sent to the chat initiator when an agent accepts a chat in response to a chat request that does not include a question. If the chat request includes a question, the agent's Answer Question text is sent in response instead of the Accept response.
- *Answer Question:* This response is automatically sent to the chat initiator when an agent accepts a chat in response to a chat request that includes a question.
- *Deny:* Only applies to collaborative chat. If an agent elects not to accept a chat, this response is automatically sent to the chat initiator.
- *End:* If either party exits a chat after the chat is accepted, this response is displayed from the agent.
- *Forward:* If the agent forwards a chat session to another queue, this response is sent to the customer.
- *Other:* These responses are never automatically sent in a chat session. Their message names appear in the Template Messages drop-down list box on the agent chat page. These messages are appended to the template messages (chat responses) defined for the queue.

**Response Name**

This name appears in the agent's template response drop-down list box.

**Response Text**

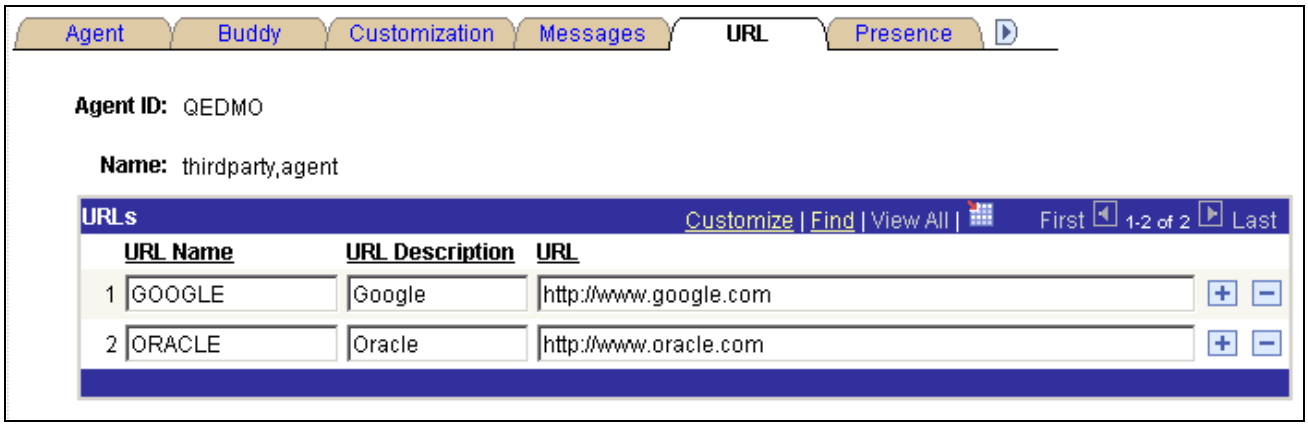
Enter the response text to appear in the chat window.

## Specifying Agent-Specific URLs

This page defines URLs that this agent can send to a client browser. Access the URL page using the following navigation path:

PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, URL


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The URL page displaying the Agent ID and Name and having the following editable fields: URL Name, URL Description, and URL.

**URL Name** The URL name appears in the agent's URL drop-down list box.

**URL Description** This description appears only on this page, to further describe this URL or, for example, its reason for inclusion.

**URL** Enter the URL. The URL must include the protocol (http://) and any required parameters.

All URLs defined for the agent are downloaded when the agent launches the agent chat console by accepting a customer chat. These URLs are not available in collaborative chat.

If you send a PeopleSoft Pure Internet Architecture URL, be sure that the recipient has permissions to access that portal, node, or page.

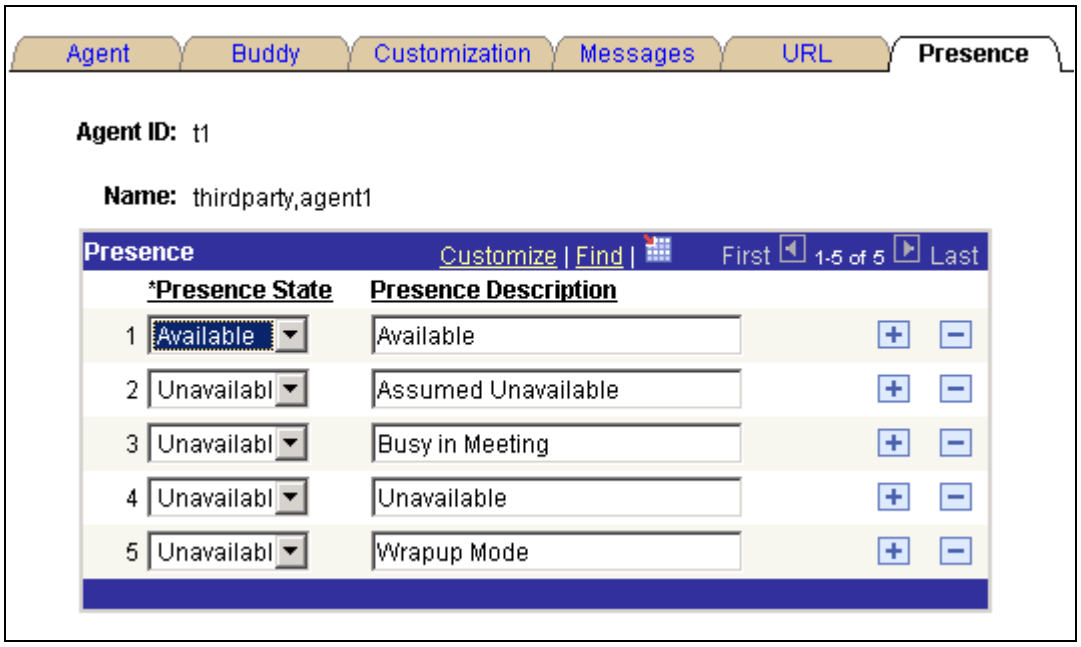
## Defining Agent's Presence

Each agent can customize the presence description displayed when the agent is available or unavailable. The queue server only understands the presence state, available or unavailable, but you can enter more specific presence descriptions when displaying or logging an agent's presence. If you do not enter presence descriptions, default values are used.

Access the Presence page using the following navigation path:

PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Presence

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Presence page displaying the Agent ID and Name and having the following editable fields: Presence State and Presence Description.

**Presence State** Select *Available* or *Unavailable.*

**Presence Description** Enter a description for each agent state. The description appears in logs of agent activity and when agent presence is displayed.

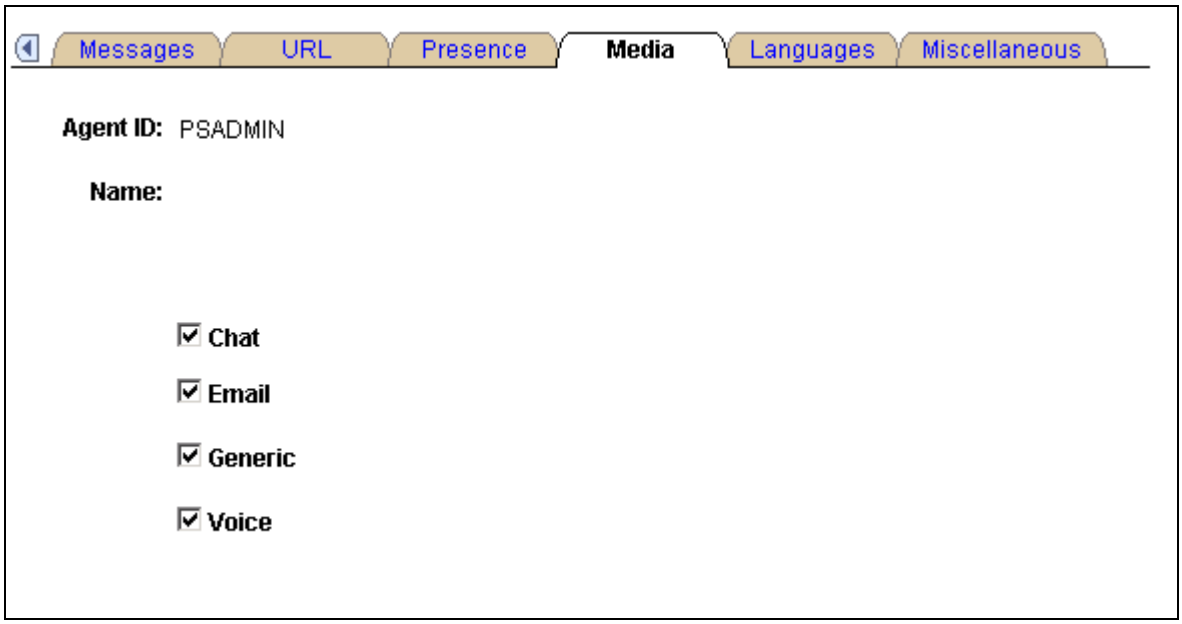
**Note.** An available state has only one presence description, *Available.* For an unavailable state, you can enter any presence description, such as tea break, coffee break, lunch, and so on.

## Specifying the Media

The media page defines the capability of each agent to perform a task. Access the Media page using the following navigation path:

PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Media


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Media page displaying the Agent ID and Name and having the Chat, Email, Generic, and Voice check boxes.

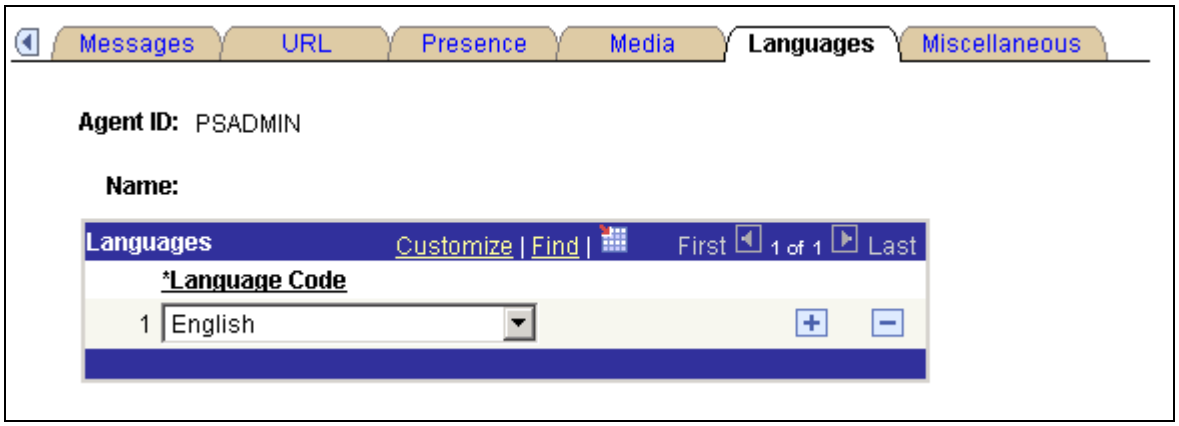
| | |
|---|---|
| **Chat** | Select this check box if the agent has a capability to chat. |
| **Email** | Select this check box if the agent has a capability to email. |
| **Generic** | Select this check box if the agent has a capability to perform a generic task. |
| **Voice** | Select this check box if the agent has a capability to perform a voice task. |

## Specifying Languages That an Agent Supports

Access the Languages page using the following navigation path:

PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Languages

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




The Languages page displaying the Agent ID and Name and having the Language Code editable field.

Enter the languages that each agent is qualified to support.

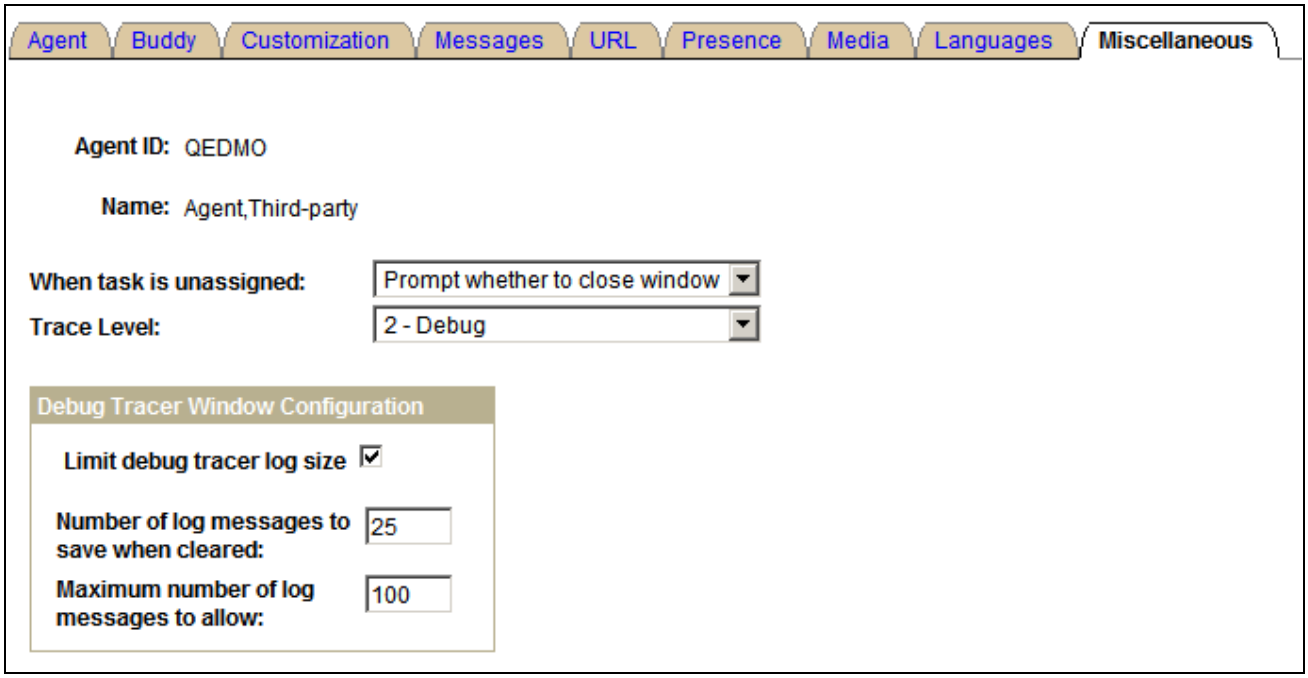
The agent is assigned only tasks that have been enqueued with a language code in the agent's language list. For the EnQueue built-in function, the language code is specified as a parameter. For the InitChat built-in function, the language code is determined by the user profile of the initiator.

If you do not enter a language code for a new agent, the default value is English.

## Specifying Miscellaneous Parameters

Access the Miscellaneous page using the following navigation path:

PeopleTools, MultiChannel Framework, Third Party Configuration, Agent, Miscellaneous




Third-party Agent - Miscellaneous page


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**When task is unassigned** Select from the following options the action that occurs when a task assigned to an agent is unassigned:

- *Prompt whether to close window (default).*
- *Close the task window.*
- *Do not close the task window.*

**Trace Level** Select from the following log trace levels:

- *0 - None*
- *1 - Information*
- *2 - Debug*

> **Note.** If a value other than *0* is selected, a tracer window appears to display activities and events on the chat or MultiChannel Console for debugging purposes.

**Limit debug tracer log size** This check box is enabled when the value entered for Trace Level is not 0. Select this check box to enable the agent to clear the tracer log based on Number of log messages to save when cleared and Maximum number of log messages to allow.

If the check box is cleared, the tracer log will not be cleared and the Number of log messages to save when cleared and Maximum number of log messages to allow fields will be disabled.

**Number of log messages to save when cleared** Specify the minimum number of recent tracer log messages that should be maintained in the tracer window.

**Maximum number of log messages to allow** Specify the maximum number of tracer log messages that will be maintained in the tracer window.

**Note.** Number of log messages to save when cleared and Maximum number of log messages to allow fields are required if the Limit debug tracer log size check box is selected.

# Configuring PeopleSoft MCF Tasks for a Third Party

This section discusses how to configure tasks. To configure tasks, use the MCF Task (MCF_TP_TASKCFG_CMP) component.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Page Used to Configure PeopleSoft MCF Task for a Third Party

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Task Configuration | MCF_TASKCFG_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Task | Configure tasks such as chat, email, voice, and generic alerts. |

### Configuring PeopleSoft MCF Task for a Third Party

See Chapter 7, "Configuring PeopleSoft MCF Queues and Tasks," Configuring Tasks, page 102.

## Configuring CTI for a Third Party

This section discusses how to configure CTI using third-party routing system. To configure CTI, use the MCF_TP_CTI_CONFIG component.

See Chapter 3, "Configuring PeopleSoft Computer Telephony Integration," Configuring PeopleSoft CTI, page 22.

## Communicating with Customers and Agents Using Chat

This section discusses how to use the Agent Chat window.

- Use the third-party agent chat window.
- Use the customer chat window.

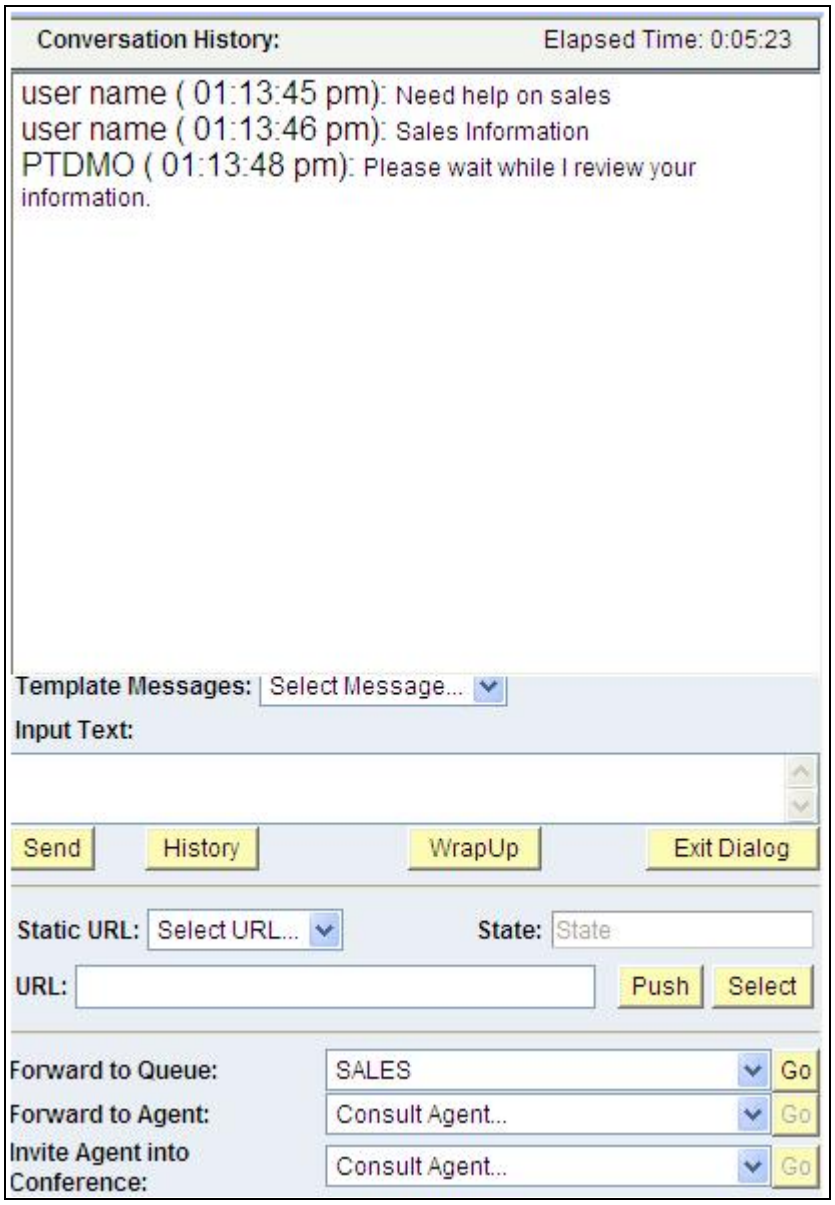
### Using the Third-Party Chat Window

When you accept a chat task, a chat window opens.

The format of the agent chat window and the contents of its right pane are delivered as part of PeopleSoft MultiChannel Framework. The content of the left pane is determined by application developers, and is passed as a parameter to the InitChat() built-in function.

The following example shows an third-party agent chat window:

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Third-party agent chat window

**Conversation History** Lists progress of the chat, line by line.

If chat logging is enabled, the conversation is recorded in the database chat log by the MCF log server. View the chat log on the Chat Log page.

If accessibility features are not turned off in the My Personalizations component, an additional text box appears below the conversation history. The most recent customer input appears in the secondary text box, which can be read by screen reading software.

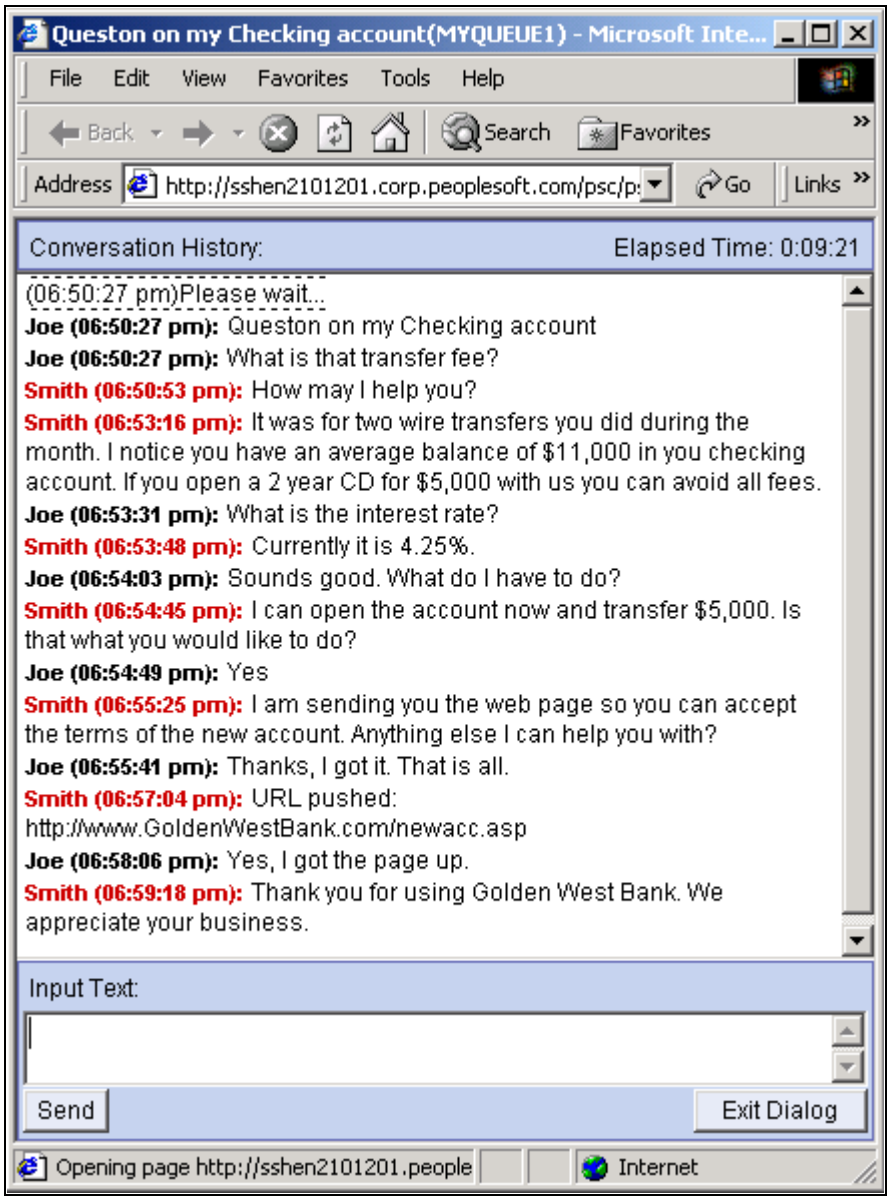
Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| | |
|---|---|
| **Template Messages** | Send the customer a standard message by selecting one from the list.<br>The message text appears in the Input Text text box. Click Send to send the message.<br>Edit template messages for each queue on the queue Chat Responses page. |
| **Input Text** | To respond to the customer, enter text, and then click Send or press Enter.<br>The maximum size of the text that can be sent at one time is 4096 bytes (4 kilobytes). |
| **Send** | Click to send the contents of the Input Text text box. |
| **Exit Dialog** | Click to end the chat.<br>The chat window remains open, enabling you to return to the page for follow-up work after the customer exits the dialog. |
| **Static URL** and **Push** | To send a static URL to the customer, select a URL name, then click Push.<br>When a URL is pushed to a customer, a browser window for that URL is launched on the customer's workstation. |
| **URL** | Displays static and grabbed URLs. |
| **Select** | Click to launch an application page, from which a URL can be returned to populate the URL field.<br>The format of the URL wizard page that appears when you click Grab is defined by application developers. PeopleSoft supplies a sample URL wizard page.<br>See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the URL Wizard, page 172. |
| **Forward to Queue** | To forward the current chat to another queue, select the queue. |
| **History** | Click to get the history for the chat conversation. |
| **Wrapup** | Click to write wrapup comments for the chat.<br>Wrapup comments are stored in PS_MCFCHATLOG. |
| **Forward to Agent** | To forward the current chat to another agent, select the queue and click Go. |
| **Invite Agent into Conference** | To request that another agent join the chat, select the agent and click Go. |

## Using the Customer Chat Window

When a customer initiates a chat, a chat window appears:


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




A Customer chat window showing the Conversation History and the Input Text

The format and content of the customer chat window is delivered as part of PeopleSoft MultiChannel Framework. The customer chat window includes a conversation history text box, input text box, and Send and Exit Dialog buttons.

If accessibility features are not turned off in the My Personalizations component, an additional text box appears below the conversation history. The most recent agent input appears in the secondary text box, which can be read by screen reading software.

**Input Text** The customer enters text here.

The maximum size of the text that can be sent at one time is 4096 bytes (4 kilobytes).

**Send** Click to send the input text to the agent.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Exit Dialog** Click to end the chat and close the chat window.

**Note.** The agent collaborative chat window is substantially the same as the customer chat window.

# Viewing Event Logs for a Third Party

This section discusses how to view event logs. To view event logs, use the (MCF_TP_EVNTLOG_CMP) component.

| Domain | Time event logged to DBMS | Event type | Task Type | Queue ID | Agent ID | CTI Agent ID | ANI | DNIS | This DN | Other DN | Call ID |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MCF1 | 12/07/2005 5:02:30PM | Restrt Ack | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/12/2005 12:09:40PM | User RQ | Voice | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/12/2005 12:09:40PM | User Event | Voice | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/15/2005 2:01:29PM | Q Shutdown | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/15/2005 2:01:29PM | Q Shutdown | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/15/2005 2:01:34PM | Restrt Ack | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/19/2005 1:33:14PM | Q Shutdown | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/19/2005 1:33:14PM | Q Shutdown | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/19/2005 1:34:02PM | Restrt Ack | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/19/2005 2:23:06PM | Accept | Generic | MARKETING | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/19/2005 2:23:06PM | DB Cntct | Generic | MARKETING | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/21/2005 11:05:00AM | Q Shutdown | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/21/2005 11:05:00AM | Q Shutdown | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |
| MCF1 | 12/28/2005 11:28:28AM | Q Shutdown | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) | (blank) |

The Event Log page displaying the Domain, Time Event logged to DBMS, Event Type, Task Type, Queue ID, Agent ID, CTI Agent ID, ANI, DNIS, This DN, Other DN, and Call ID columns.

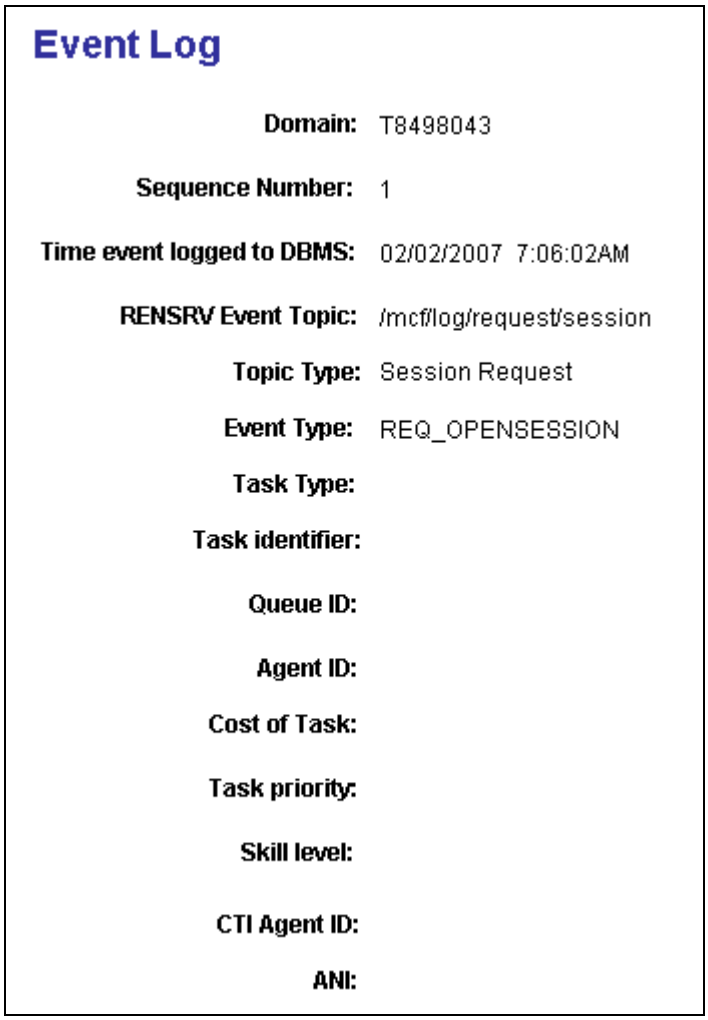
## Page Used to View Event Logs for a Third Party

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Event Log | MCFTPEVENTLOG_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Event Log | Displays the details of any event. |

## Viewing Event Logs for a Third Party

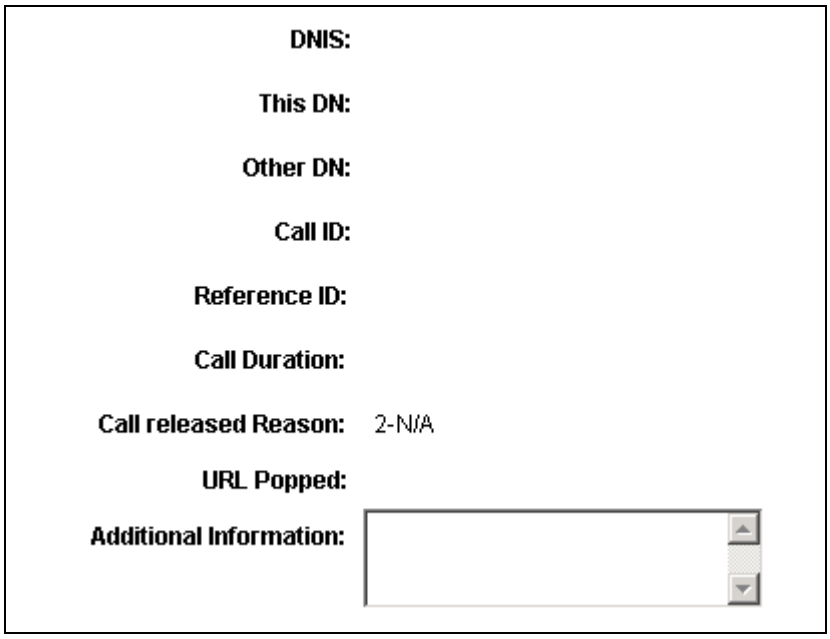
Select any event from the Event Log page to display the details of any event.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Event Log page 1 of 2 displaying the Domain, Sequence Number, Time Event Logged to DBMS, RENSRV Event Topic, Topic Type, Event Type, Task Type, Task Identifier, Queue ID, Agent ID, Cost of Task, Task Priority, Skill Level, CTI Agent ID, and ANI fields.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Event Log page 2 of 2 displaying the DNIS, This DN, Other DN, Call ID, Reference ID, Call Duration, Call Released Reason, URL Popped, and Additional Information fields.

# Viewing Broadcast Logs for a Third Party

To view the broadcast log, access the broadcast log (MCF_TP_BCASTLG_CMP) component.

## Page Used to View Broadcast Logs for a Third Party

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Broadcast Log | MCF_BCASTLOG_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Broadcast Log | View broadcast logs. |

## Viewing Broadcast Logs for a Third Party

See Chapter 9, "Administering Queues, Logs, and Tasks," Viewing Broadcast Logs, page 141.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Working with Third-Party Sample Pages

To demonstrate tools and functionality, use the Sample Pages (MCF_TP_DEMO_CMP) component. These sample pages demonstrate how JSMCAPI can be used with the PeopleCode behind it. The third party may develop their own interface, which may look significantly different from the sample pages included in the section.

This section discusses how to:

- Use the Customer Chat sample page.
- Use the Generic Event sample page.
- Use the Email sample page.
- Use the Sample Console page.
- Use the JSMCAPI Broadcast page.
- Use the PeopleCode Broadcast page.

## Pages Used to Work with Third-Party Sample Pages

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Customer Chat Page | MCFTP_DEMO_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Sample Pages, Customer Chat Page | Send a customer chat session and use built-in functions. |
| Generic Page | MCFTP_DEMO_NTFY_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Sample Pages, Generic Page | Send a generic event notification and use built-in functions. |
| Email Page | MCFTP_DEMOEM_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Sample Pages, Email Page | Send an email and use built-in functions and application package classes. |
| Sample Console Page | PT_SAMPLETPAGTCNSL | PeopleTools, MultiChannel Framework, Third-Party Configuration, Sample Pages, Sample Console Page | Launch a sample MultiChannel Console generated using JSMCAPI and perform all tasks. |
| JSMCAPI Page | MCFTP_BROADCAST_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Sample Pages, JSMCAPI Page | Send a broadcast message using JSMCAPI. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Page Name | Definition Name | Navigation | Usage |
|---|---|---|---|
| PCodeBroadcast Page | MCFTP_PCODEBCST_PG | PeopleTools, MultiChannel Framework, Third-Party Configuration, Sample Pages, PCodeBroadcast Page | Send a broadcast message using PeopleCode. |

## Using the Customer Chat Sample Page

See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the Customer Chat Sample Page, page 170.

## Using the Generic Event Sample Page

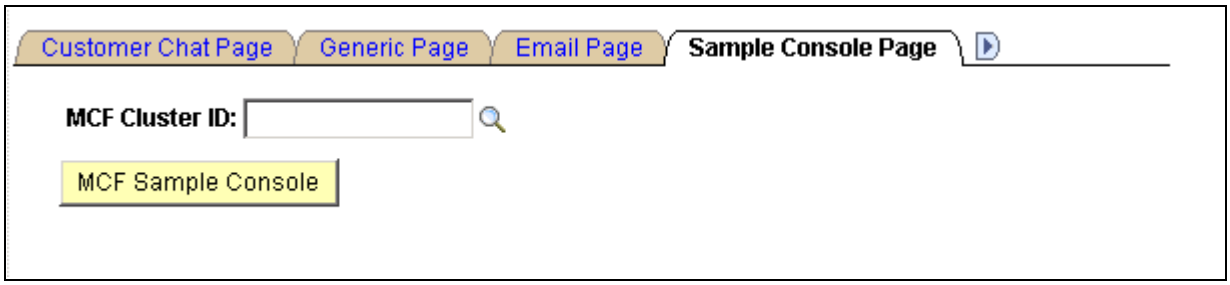
See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the Generic Event Sample Page, page 173.

## Using the Email Sample Page

See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the Email Sample Page, page 176.

## Using the Sample Console Page

Access the Sample Console page using the navigation path: PeopleTools, MultiChannel Framework, Third-Party Configuration, Sample Pages, Sample Console Page. Click the lookup button and select the REN cluster.




The Sample Console page having the MCF Cluster ID editable field and the MCF Sample Console button.

**MCF Sample Console** Click to initiate the sample console.

**Note.** To demonstrate the MCF sample console, you must have an MCF cluster running and communicating with the specified REN server cluster.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

After clicking MCF Sample Console, a new browser window displaying the sample console appears along with a tracer window. The Sample Console contains the following group boxes, which are detailed in the following topics:

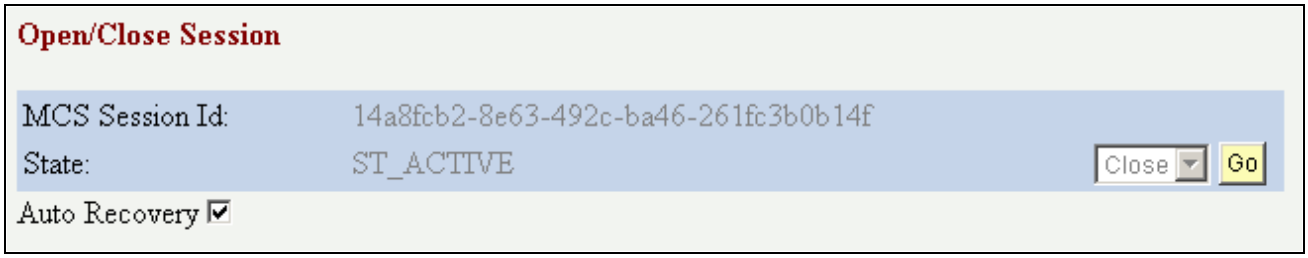
- Open/Close Session
- Register User to Session
- Register Group, Login User to Group, and User-Group States
- Address Operations
- Register Extension to Session, and Extension Operations
- Buddies
- Buddies Bulk Registration
- Groups Bulk Registration
- Error Messages / Information
- Server State and Broadcasts

**Note.** To enable the new browser window to appear, disable any pop-up blocking software for your browser.

Before accessing the Open/Close Session group box, enable the third-party routing server.

### *Open/Close Session*

Access the Open/Close Session group box. (Select PeopleTools, MultiChannel framework, Third-party Configuration, Sample Pages. Enter the MCF Cluster ID and click the Sample Console button).




The Open/Close Session group box

**MCS Session Id** Displays the session ID of the MultiChannel server.

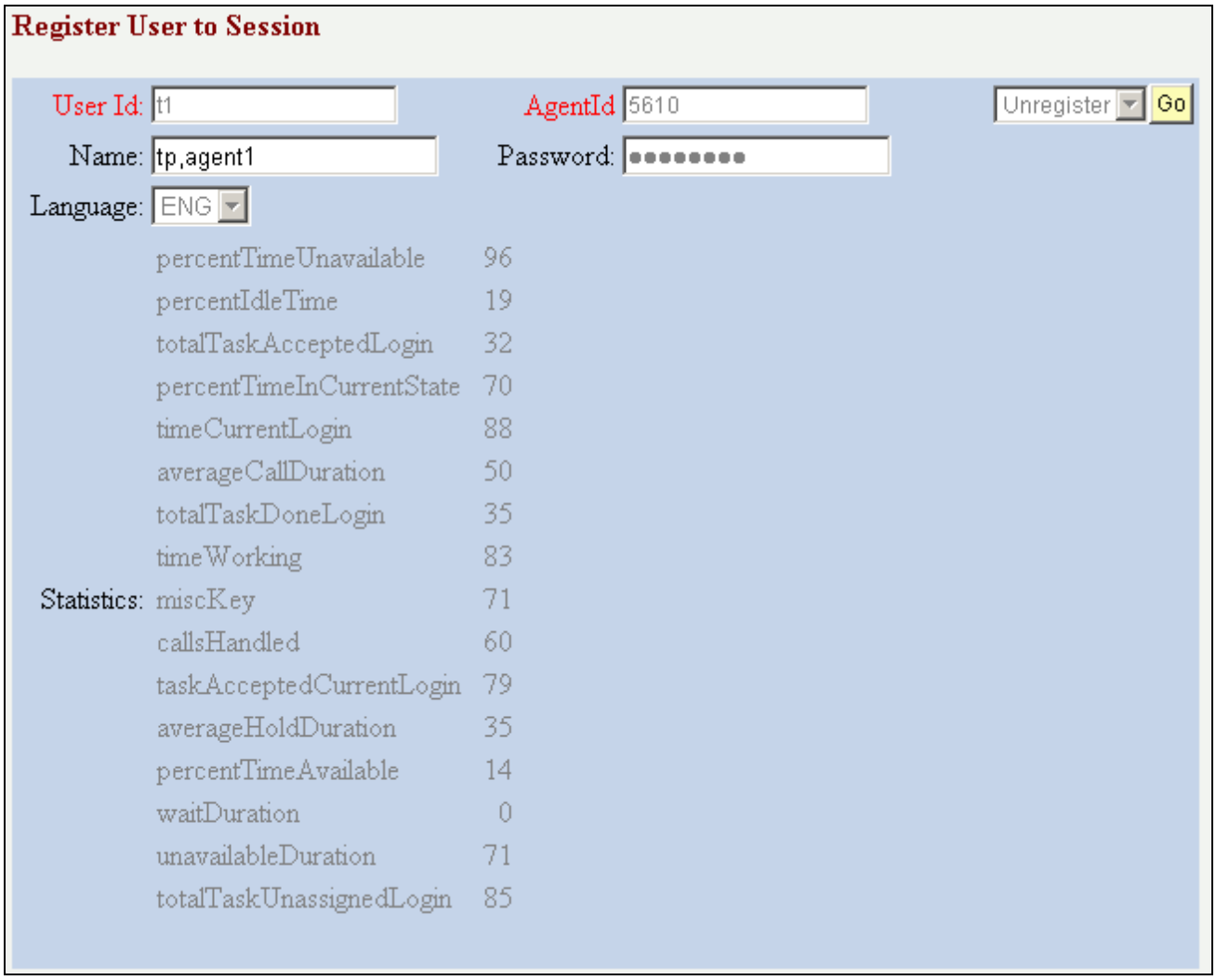
**State** Displays the state of the session.

**Note.** If the state is *ST_FAILED*, check the third-party routing server. It must be up and running.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Register User to Session*

Access the Register User to Session group box. (Select PeopleTools, MultiChannel framework, Third-party Configuration, Sample Pages. Enter the MCF Cluster ID and click the Sample Console button).




The Register User to Session group box

Register a user to the session in the Register User to Session group box.

**User Id** Enter user ID of the agent.

**Agent Id** Enter agent's ID.

**Note.** Because a CTI agent can have an agent ID different from the agent's user ID, both values are required. If the agent has a password, the password is also required.

**Name** Enter the name of the agent.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Password** Enter the password if the agent has a password.

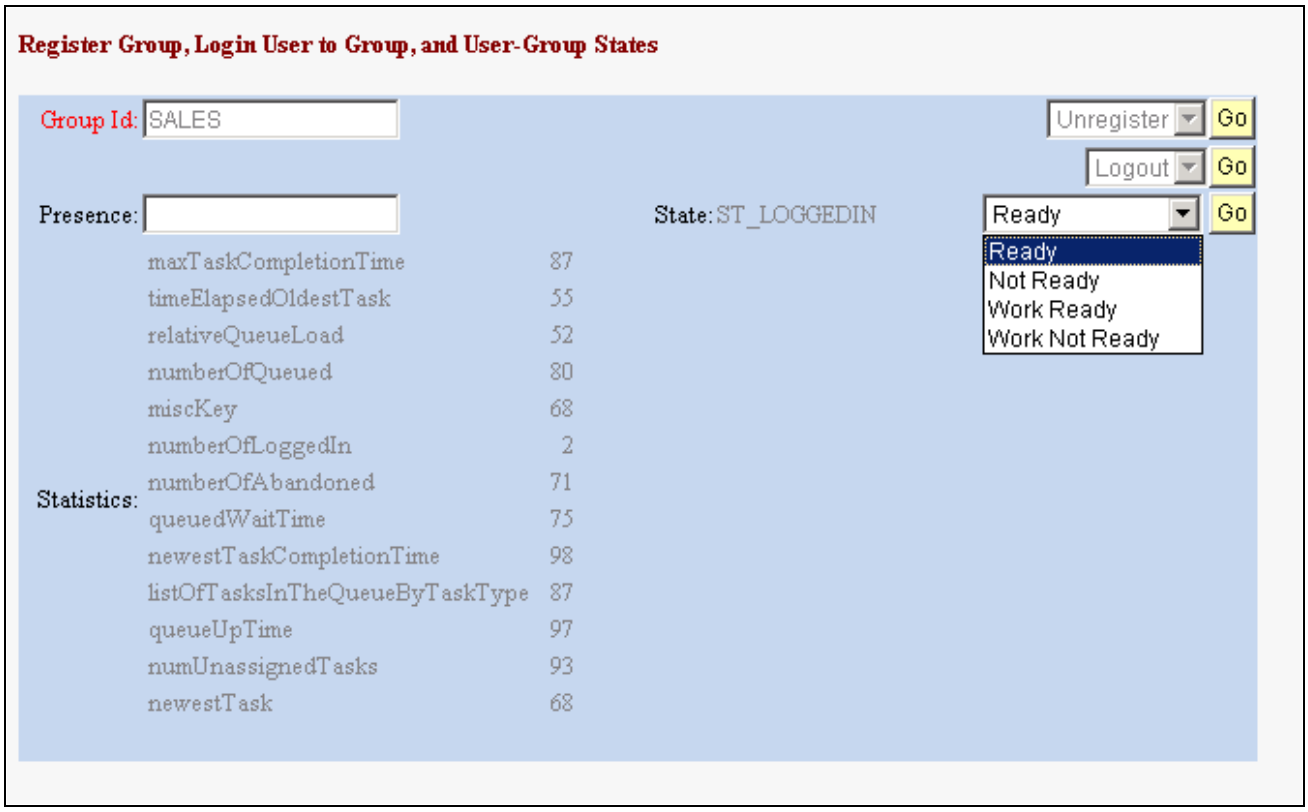
**Language** Select the language.

The language field is reserved for future use. The default language is English.

**Statistics** Displays the agent statistics.

### *Register Group, Login User to Group, and User-Group States*

Access the Register Group, Login User to Group, and User-Group States group box. (Select PeopleTools, MultiChannel framework, Third-party Configuration, Sample Pages. Enter the MCF Cluster ID and click the Sample Console button).




The Register Group, Login User to Group, and User-Group States group box

**Group Id** Enter the name of the group to which the agent will be registered. The group ID is the same as the name of the queue to which the agent is logged in.

To register the agent with the specified group, click the Go button next to the Register field.

**Note.** After you click Go, the field value changes to *Unregister*.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Login** Click the Go button to log in if *Login* appears in the drop-down list box.

Click the Go button to log out if *Logout* appears in the drop-down list box.

---

**Note.** After you log in to the group, a new pop-up window opens, prompting you for reason code data. If you register to an invalid group, a pop-up window opens showing the error messages.

---

See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Understanding JSMCAPI, page 178.

**Presence** Enter a presence description to associate with the state selected in the drop-down list box.

This presence value is different from any presence values predefined in the agent configuration. The agent cannot select from the predefined values.

**State** Select an agent state from the following options:

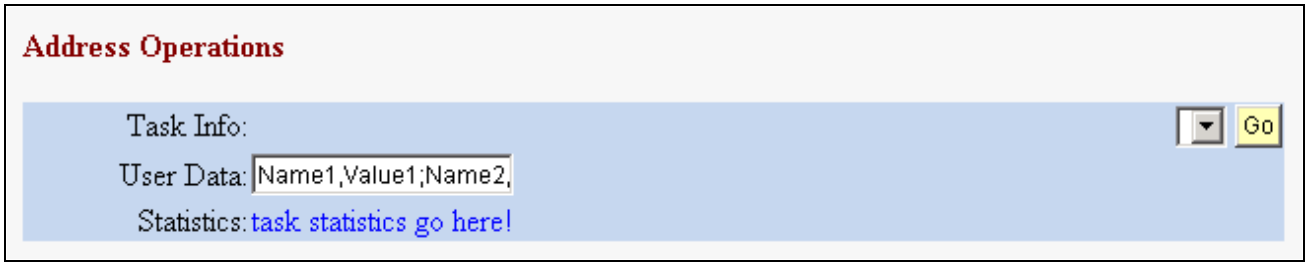
- *Ready*
- *Not Ready*
- *Work Ready*
- *Work Not Ready*

Each state value has an associated state code that appears in the State field. To enable incoming calls, select *Ready* from the drop-down list box and click the Go button.

**Statistics** Displays the group statistics.

### *Address Operations*

Access the Address Operations group box. (Select PeopleTools, MultiChannel framework, Third-party Configuration, Sample Pages. Enter the MCF Cluster ID and click the Sample Console button).




The Address Operations group box

**Task Info** (task information) Displays task information.

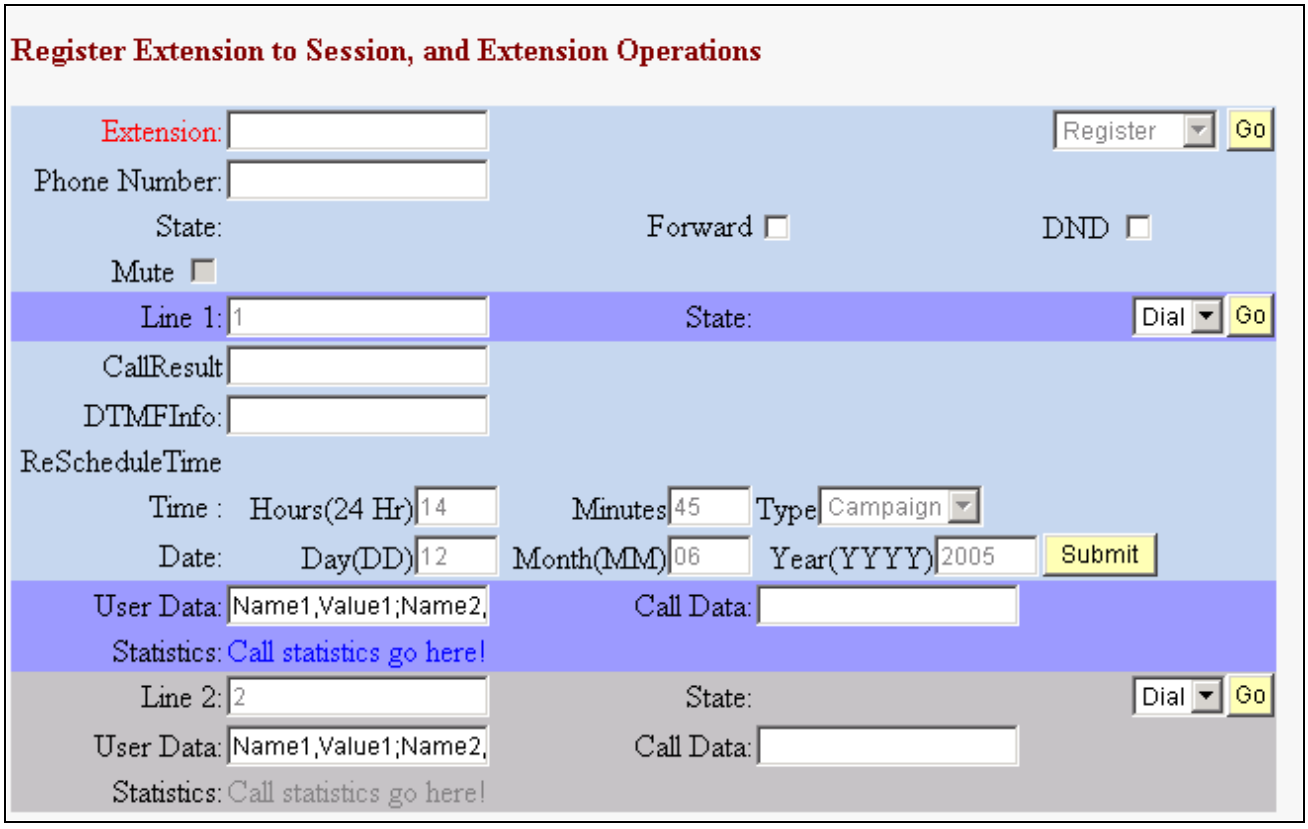
**User Data** Displays user data.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Statistics** Displays task statistics.

### *Register Extension to Session and Extension Operations*

Access the Register Extension to Session, and Extension Operations group box. (Select PeopleTools, MultiChannel framework, Third-party Configuration, Sample Pages. Enter the MCF Cluster ID and click the Sample Console button).




The Register Extension to Session, and Extension Operations group box

**Extension** Enter a valid telephone extension number.

**Phone Number** Enter a valid telephone number for use in subsequent operations, such as dial, forward, or do-not-disturb (DND).

**Mute** Select to mute the call.

**Note.** Mute is enabled only when the agent is on the call.

**Forward** Select to forward incoming calls to the specified number.

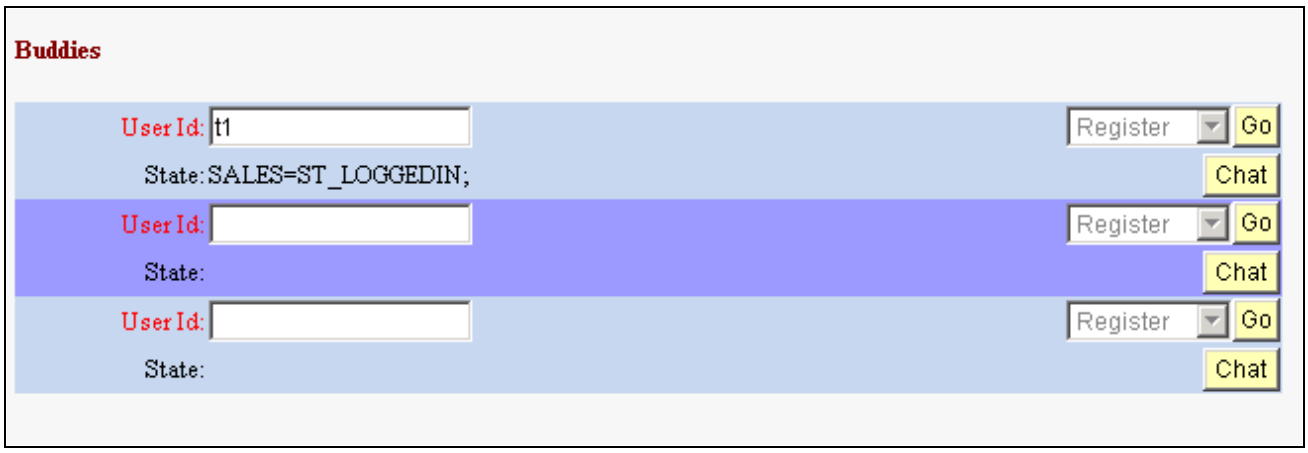
**DND**(do not disturb) Select to put the selected extension in do-not-disturb status.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| | |
|---|---|
| **Line 1** or **Line 2** | Displays a name or value for each line.<br>On the sample console, the value will either be *1* or *2*. |
| **DTMF Info** (Dual Tone Multi-Frequency information) | Select to send DTMF digits on the line. |
| **CallResult** | Select to set call result on the line. |
| **ReScheduleTime** | Select to set reschedule time on the line. |
| **Type** | Select to set reschedule type on the line. |
| **User Data** | Enter name-value pairs representing user data to be attached to a call. |
| **Call Data** | Enter any call data to be attached to the call.<br>Call data includes:<br>• *ANI*<br>• *DNIS*<br>• *THISDN*<br>• *OTHERDN* |

### *Buddies*

Access the Buddies group box. (Select PeopleTools, MultiChannel framework, Third-party Configuration, Sample Pages. Enter the MCF Cluster ID and click the Sample Console button).




The Buddies group box

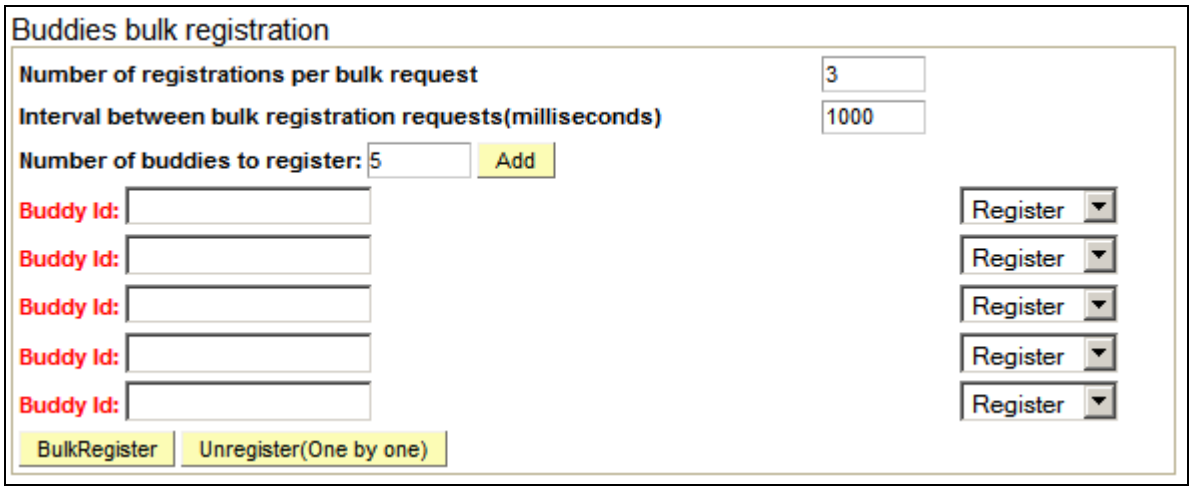
| | |
|---|---|
| **User Id** | Enter the user ID of the buddy with whom you want to chat. |
| **State** | Displays the state of the buddy. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| | |
|---|---|
| Chat | Click to initiate an agent-to-buddy chat session. |

### *Buddies Bulk Registration*

Access the Buddies bulk registration group box. (Select PeopleTools, MultiChannel framework, Third-party Configuration, Sample Pages. Enter the MCF Cluster ID and click the Sample Console button).




Buddies bulk registration group box

| | |
|---|---|
| **Number of registrations per bulk request** | Enter the number of registrations per bulk request. |
| **Interval between bulk registration requests(milliseconds)** | Enter the interval between bulk registration requests in milliseconds. |
| **Number of buddies to register** | Enter the number of buddies to add and click the Add button. Edit boxes for Buddy Id will appear. |
| **BulkRegister** | Click to register the buddies. An Unregister button will appear next to each Group Id. You can use this button to unregister a specific group. |
| **Unregister(One by One)** | Click to unregister groups one by one. |

### *Groups Bulk Registration*

Access the Groups bulk registration group box. (Select PeopleTools, MultiChannel framework, Third-party Configuration, Sample Pages. Enter the MCF Cluster ID and click the Sample Console button).

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Groups bulk registration group box

| | |
|---|---|
| **Number of registrations per bulk request** | Enter the number of registrations per bulk request. |
| **Interval between bulk registration requests(milliseconds)** | Enter the interval between bulk registration requests in milliseconds. |
| **Number of groups to register** | Enter the number of groups to add and click the Add button. Edit boxes for Group Id will appear. |
| **BulkRegister** | Click to register the buddies. An Unregister button will appear next to each Buddy ID. You can use this button to unregister a specific buddy. |
| **Unregister(One by One)** | Click to unregister buddies one by one. |

### *Error Messages/Information*

Access the Error Messages/Information group box (Select PeopleTools, MultiChannel framework, Third-party Configuration, Sample Pages. Enter the MCF Cluster ID and click the Sample Console button).

Error Messages/Information group box

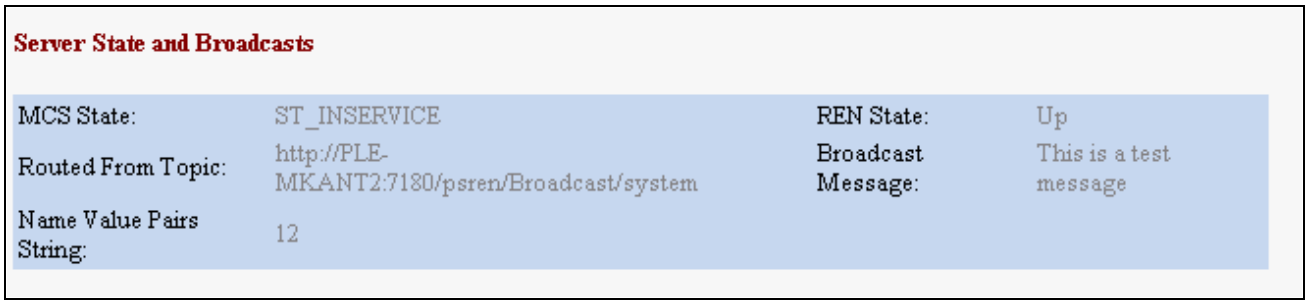
Any error messages associated with the process appear here.

| | |
|---|---|
| **Clear** | Click the button to clear a displayed error message. |

### *Server State and Broadcasts*

Access the Server State and Broadcasts group box. (Select PeopleTools, MultiChannel framework, Third-party Configuration, Sample Pages. Enter the MCF Cluster ID and click the Sample Console button).


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Server State and Broadcasts group box

**MCS State** Displays the MultiChannel server's state.

**Routed From Topic** Displays the machine name from where the broadcast message was sent.

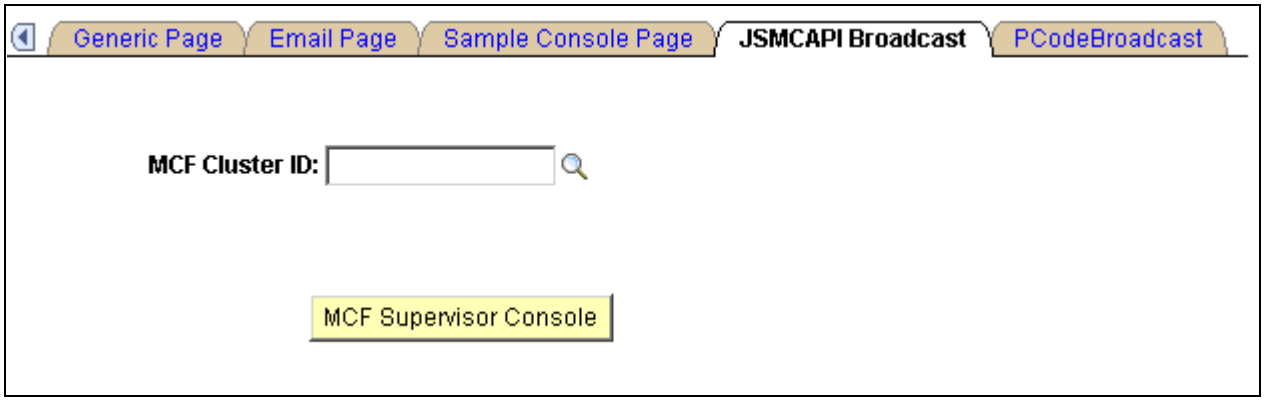
**Name Value Pairs** Displays the name value pair string.

**REN State** Displays the status of the REN server.

**Broadcast Message** Displays the supervisor's broadcast message.

## Using the MCF Broadcast Page

This section discusses how to use the JSMCAPI Broadcast page for the third party.




The JSMCAPI Broadcast page having the MCF Cluster ID editable field.

**MCF Cluster ID** Select the REN server cluster on which to test the MCF Supervisor console.

**Note.** The MCF Supervisor console is specific to the REN cluster Id selected.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**MCF Supervisor Console** Click to initiate the supervisor console.

**Note.** To demonstrate the MCF Supervisor console, you must have MCF servers, both queue and log servers, running and communicating with the specified REN server cluster.

After you click MCF Supervisor Console, a new browser window appears that displays the sample supervisor console.

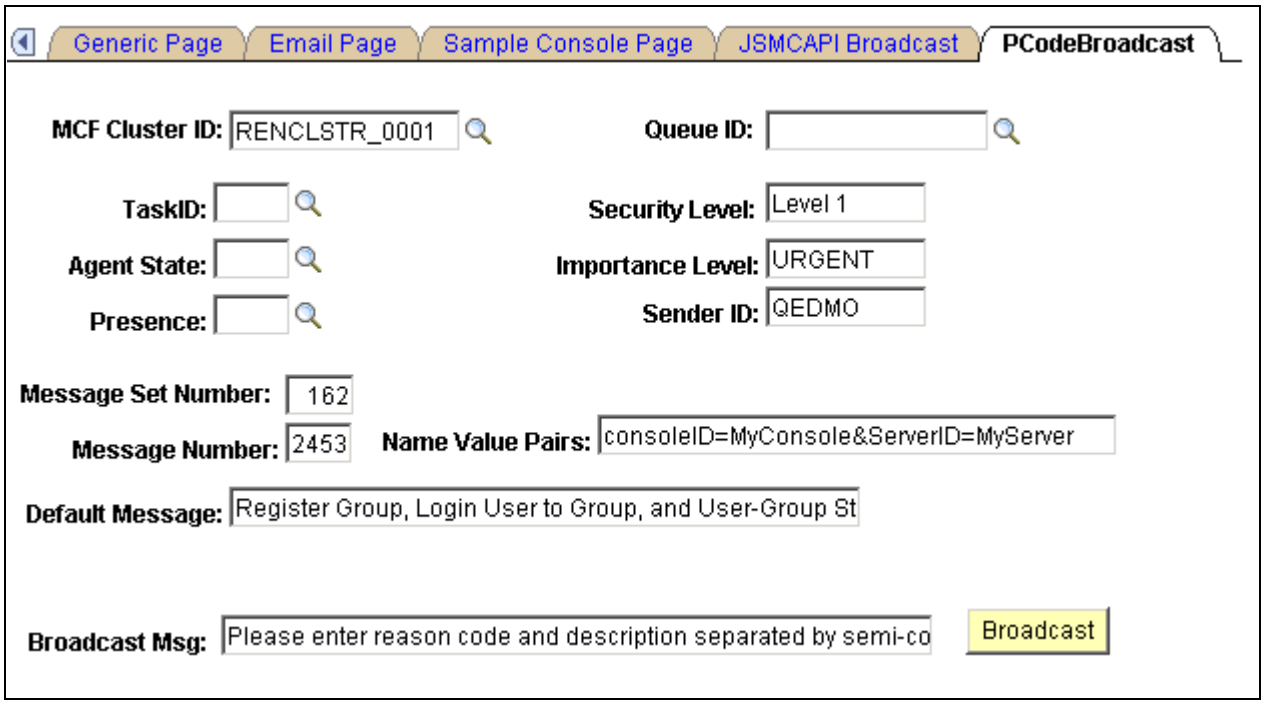
**Note.** To enable the new browser window to appear, disable any pop-up blocking software for your browser.

### *See Also*

Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using JSMCAPI Broadcast with MCF Supervisor Console, page 161

## Using the PCodeBroadcast Page

This section discusses how to use the PCodeBroadcast page.




The PCodeBroadcast Page having the following editable fields: MCF Cluster ID, Queue ID, Task ID, Security Level, Agent State, Importance Level, Presence, Sender ID, Message Set Number, Message Number, Name Value Pairs, Default Message, and Broadcast Message.

See Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using PeopleCode Broadcast, page 166.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 13

# Understanding JSMCAPI Classes

This chapter discusses the JavaScript MultiChannel Application Programming Interface (JSMCAPI) classes.

## Understanding JSMCAPI

This section discusses JSMCAPI, a JavaScript-based application programming interface (API) used to build custom applications, such as MultiChannel consoles, or to enable MultiChannel functionality on the PeopleSoft Pure Internet Architecture page. The API is built on the REN server JavaScript client and is a pure JavaScript API.

JSMCAPI is the interface through which the application developer accesses the JSMCAPI functionality. JSMCAPI provides a set of objects, such as User, Address, Group, and so on. The PSMC, a global object, provides all those methods called to send agent requests to the server; it also provides an event handler callback method for the PeopleSoft MultiChannel Application Programming Interface (PSMCAPI) call backs for events coming from the CTI server.

The JSMCAPI communicates with the PSMCAPI through the REN server using MCP (Multi-Channel Protocol). MCP includes topic and event definition. In general, JSMCAPI:

- Sends agent requests to the MultiChannel Server.

  The requests can be agent state requests such as login, logout, set ready, and so on. Requests can also be call requests, such as dial-out, answer the call, transfer/conference, and so on.

- Receives PBX/Switch events from the CTI server.

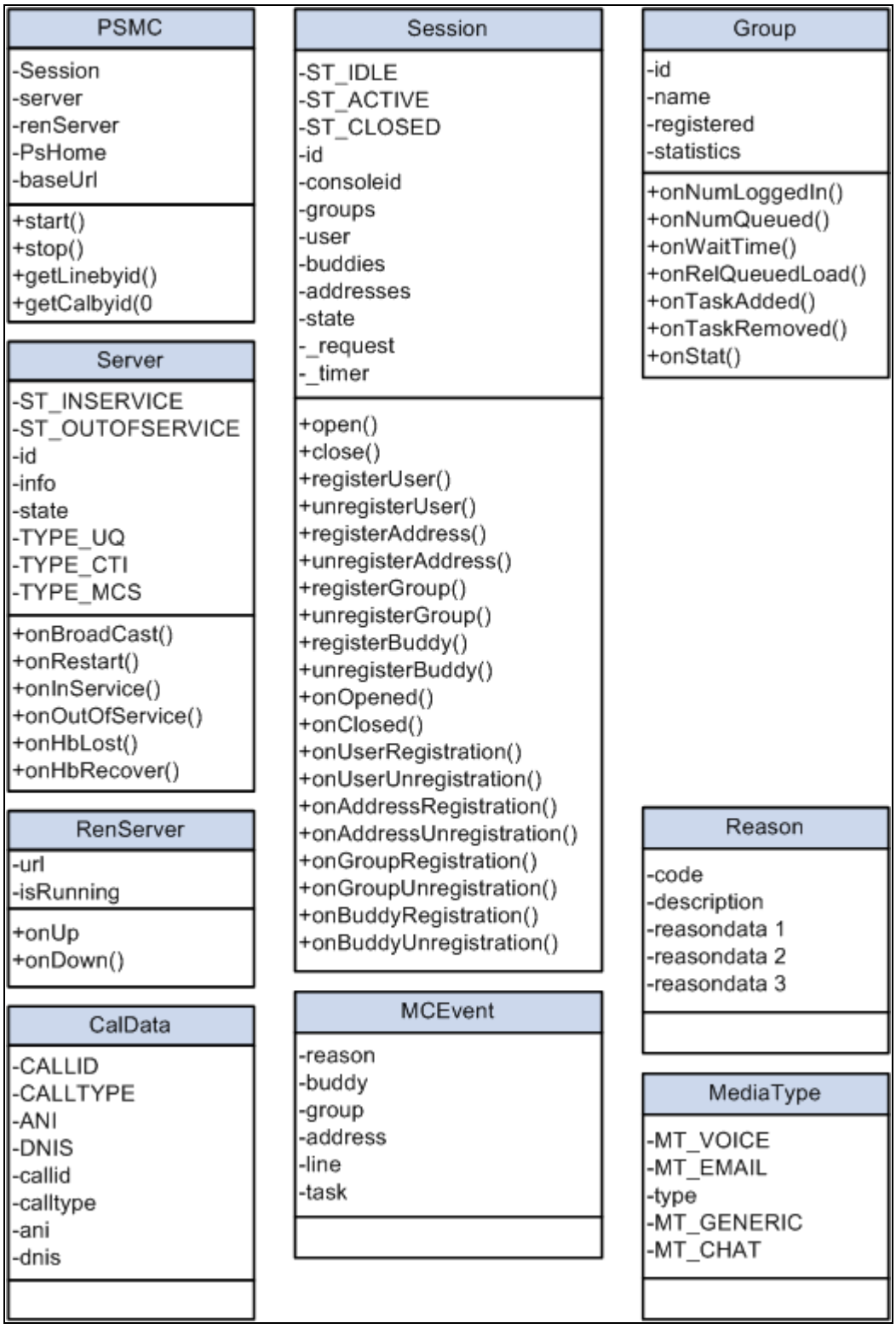
  Events can be the agent state change, such as event ready, not ready, and so on. Events can also be call events, such as incoming call, call released, call transferred, and so on.

## Understanding JSMCAPI Classes

This sections gives an overview of all the JSMCAPI classes.

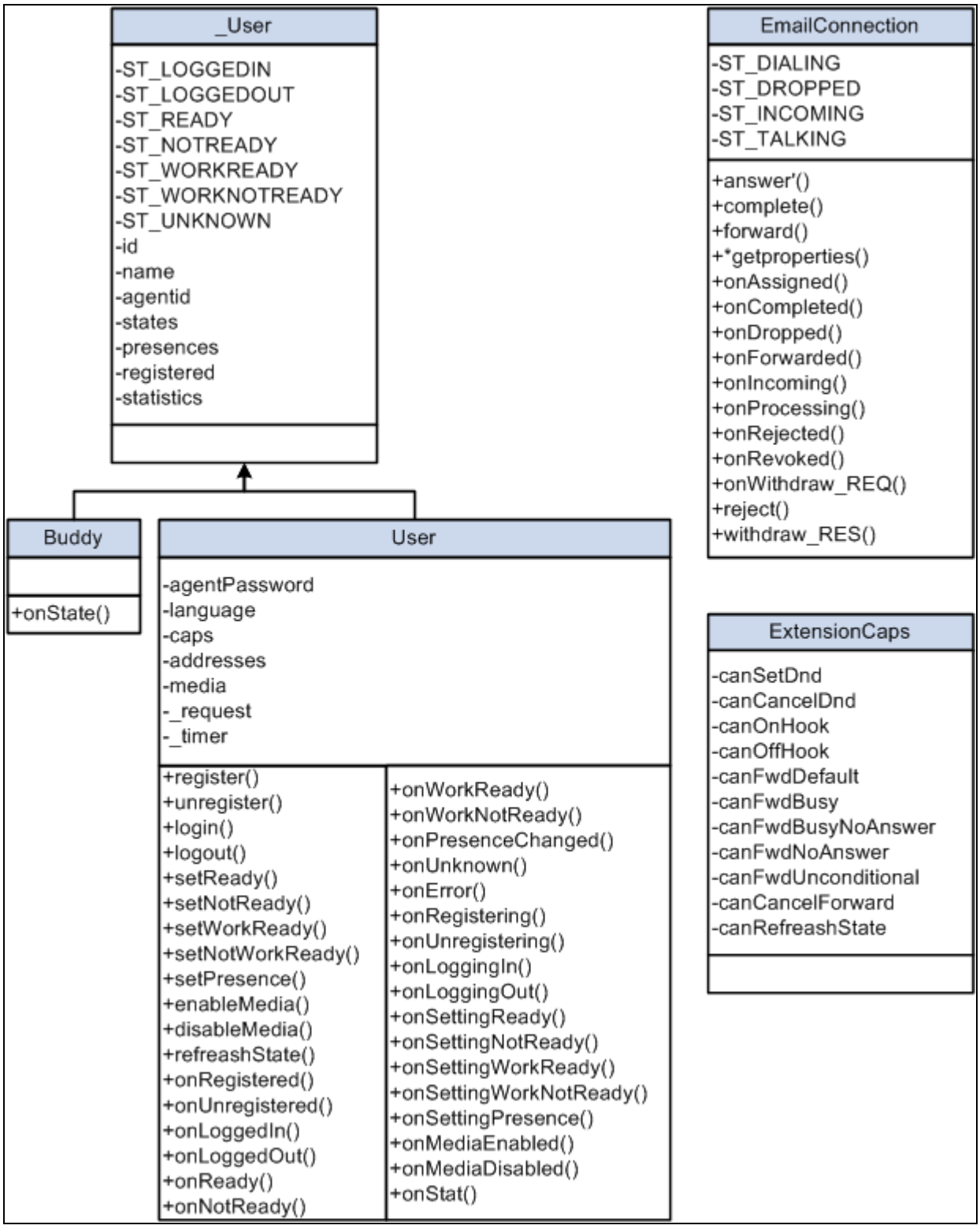
The following class diagrams explain all the classes, fields, and the methods.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.



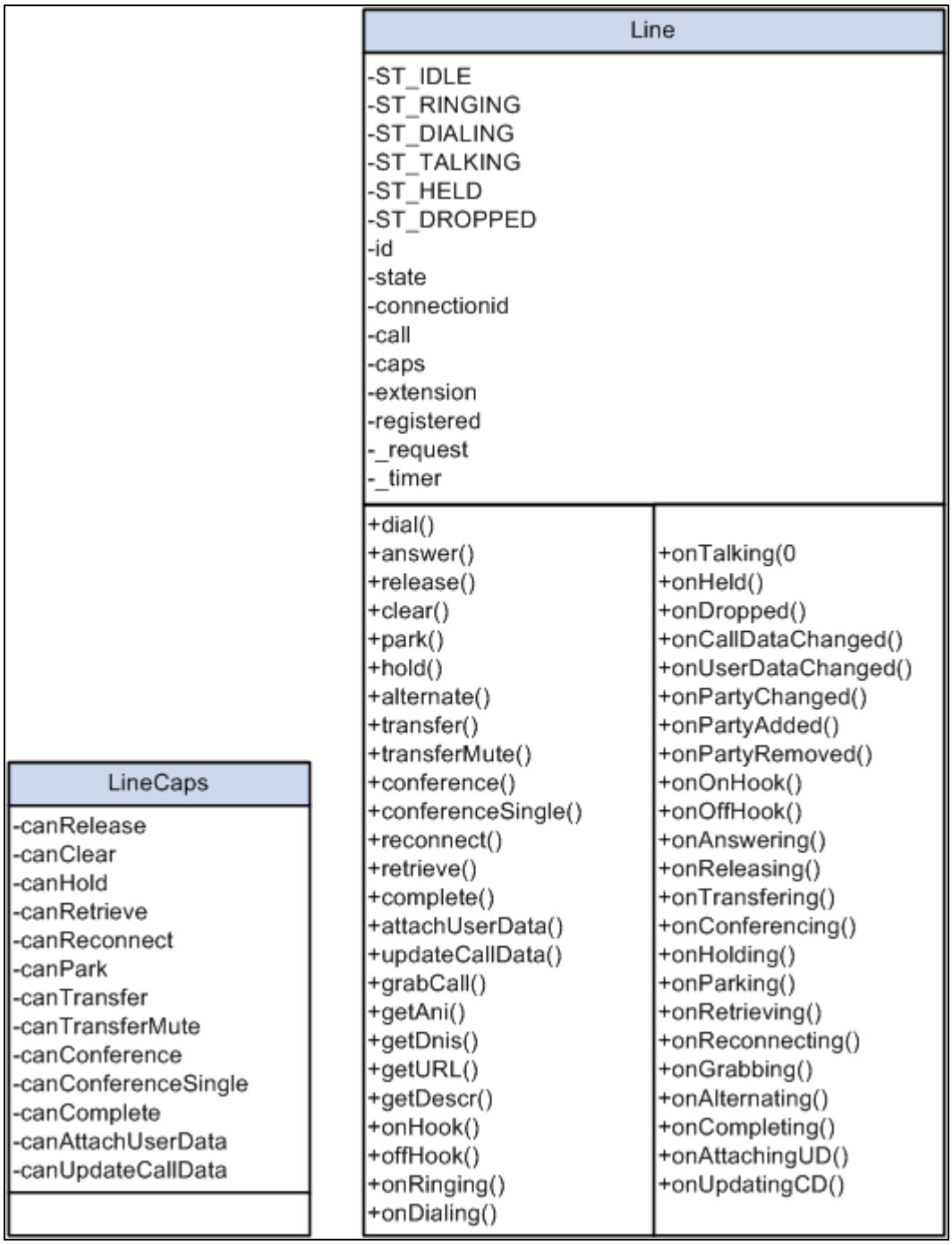

Class diagram part 1


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.



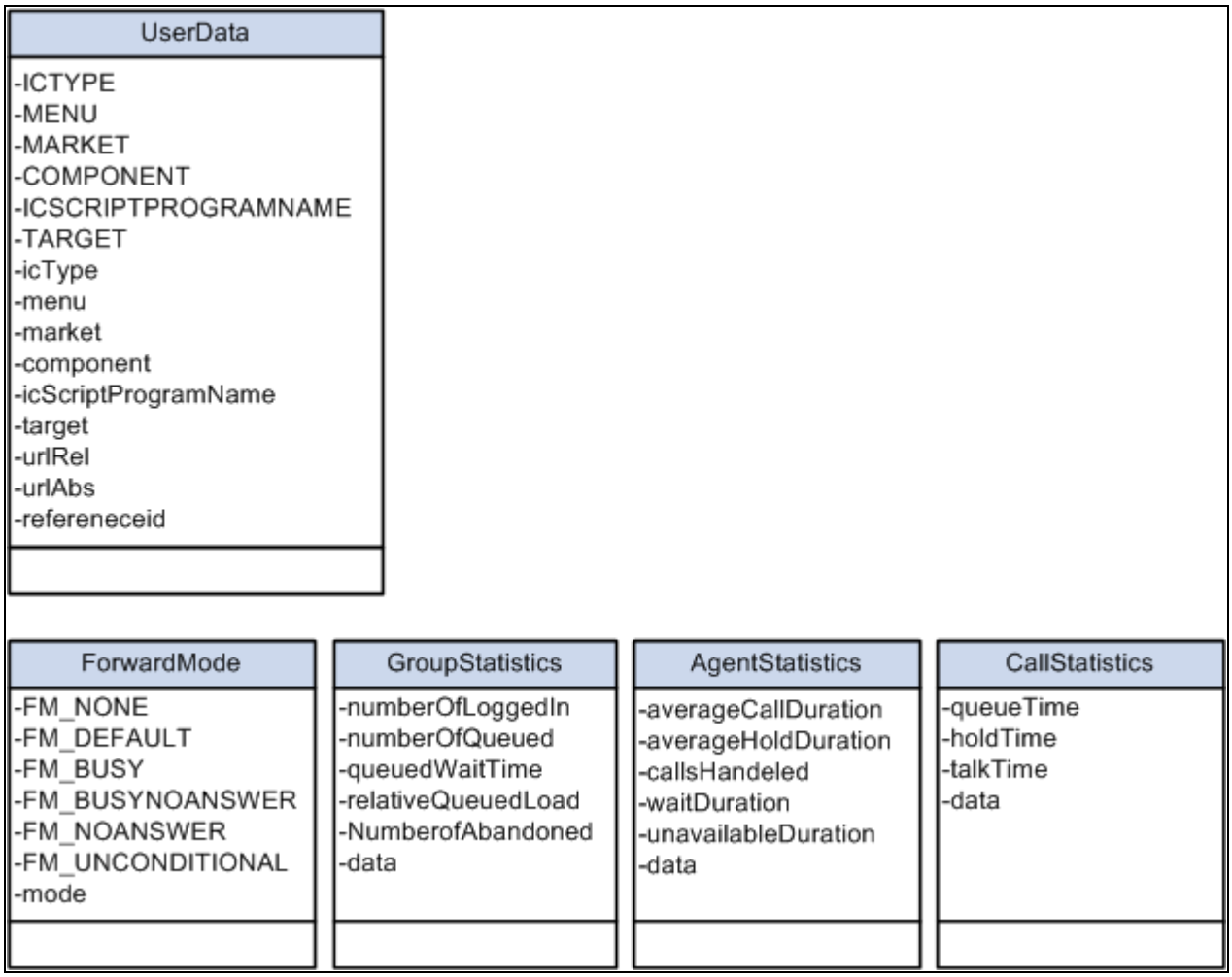

Class diagram part 2

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Class diagram part 3


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.



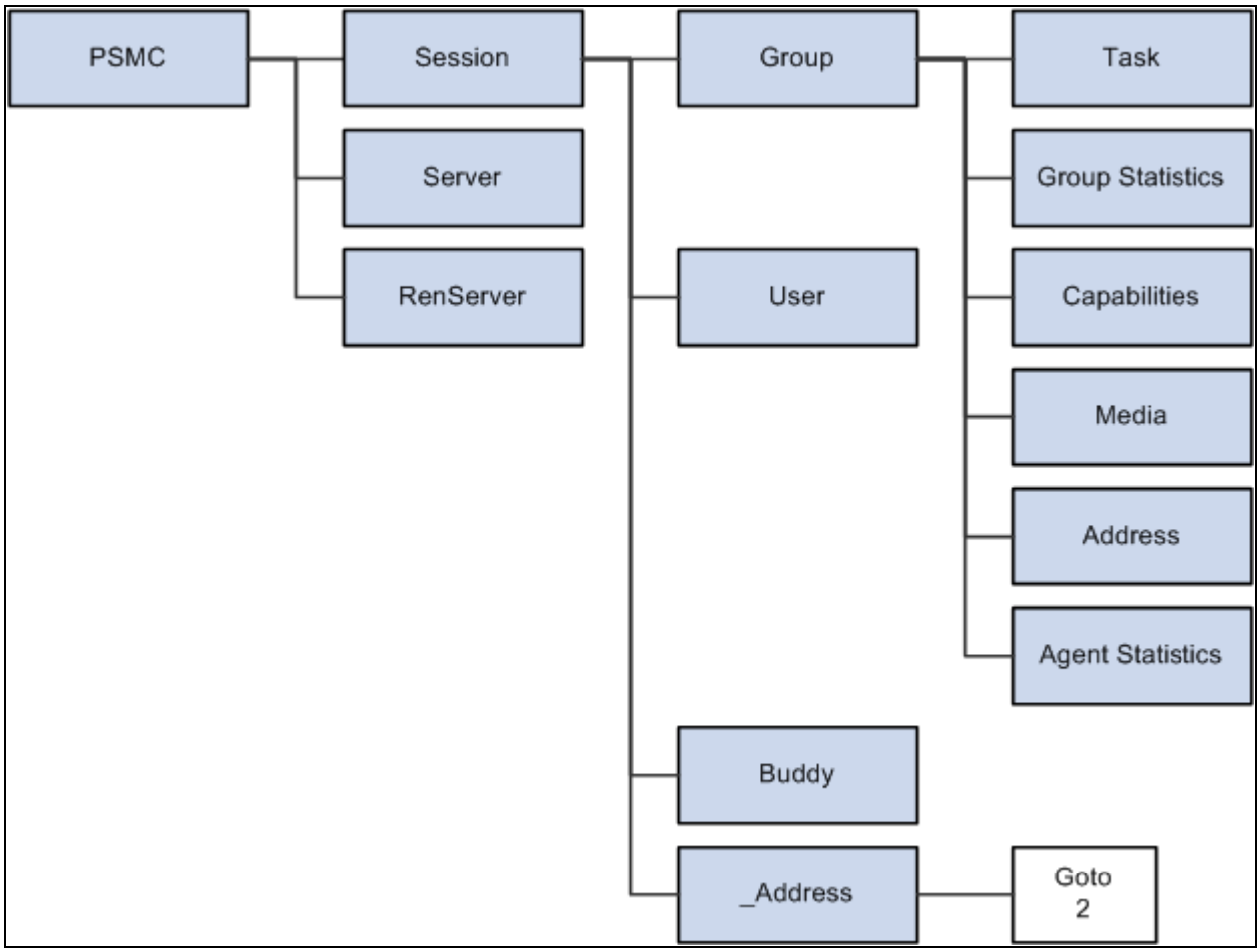

Class diagram part 4

## PSMC

PSMC is the base class that an application accesses to start an instance of JSMCAPI. The application can access all JSMCAPI functionality through this object. A session object is created from this class.

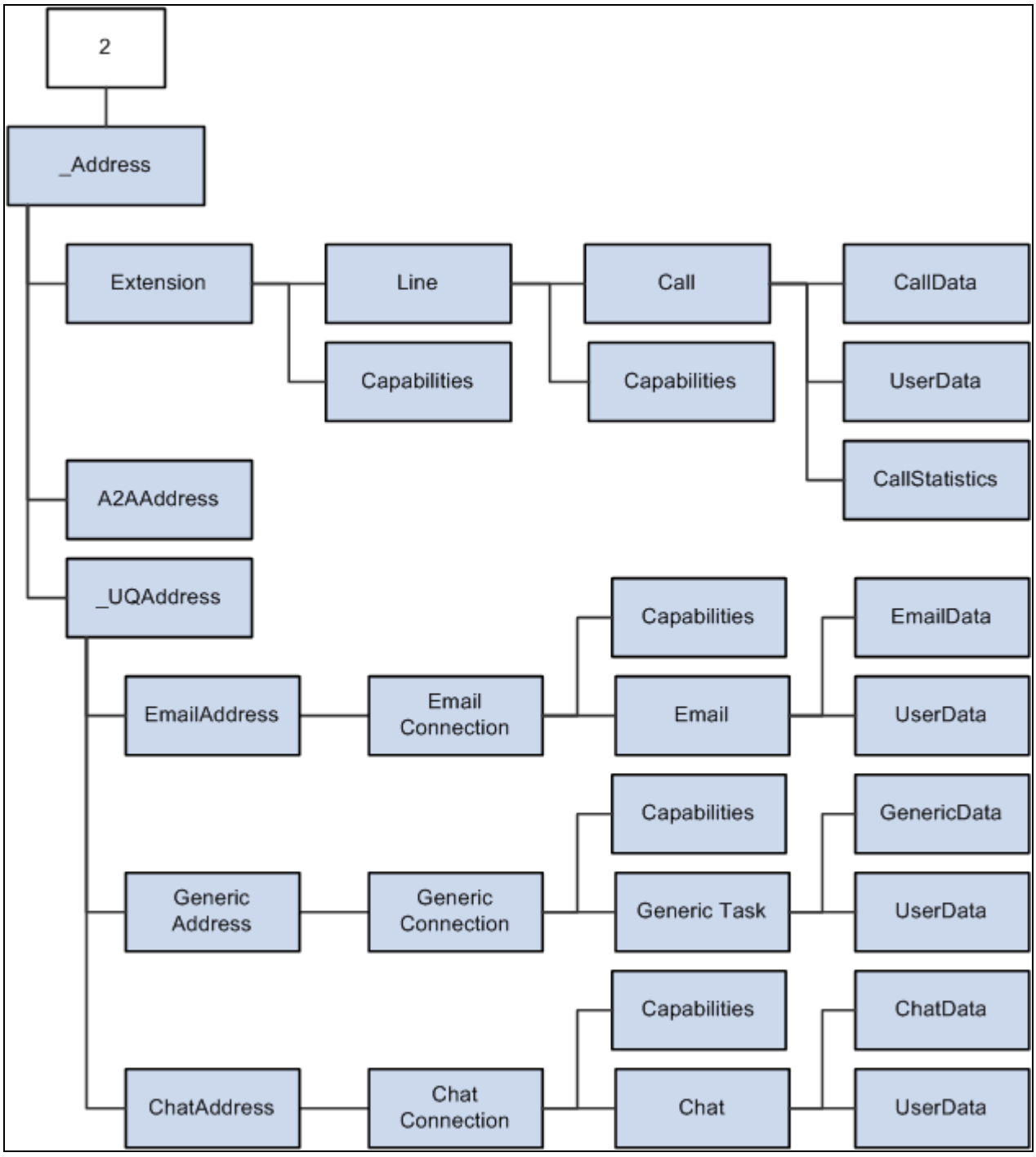
The following diagram explains the relationship of PSMC with all other classes:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 




PSMC base class part 1


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




PSMC base class part 2

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," PSMC Class Constructor, page 435

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Server

The Server class refers to the routing server and can execute events for specific server states. Three types of servers are available: CTI, queue server, and MultiChannel server. The constants are CTI, UQ and MCS, respectively.

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Server Class Constructor, page 446

## RENServer

The RENServer cluster is represented in the RENServer class. It provides the URL and the server status (active or shutdown).

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," RenServer Class Constructor, page 444

## Session

Sessions are set for users registering with the server. Addresses, buddies, and groups are registered for the user with the session. The connection between the server and JSMCAPI is a session. There is only one session object per PSMC object.

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Session Class Constructor, page 450

## _Address

The _Address class identifies the user with a unique ID. The ID identifies the user to the routing system and is unique for each channel or media type.

The subclasses of _Address class are:

- Extension
- _UQAddress
- A2AchatAddress

Further delineation of address by media type is provided by:

- ChatAddress

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- EmailAddress
- GenericAddress

ChatAddress, EmailAddress, and GenericAddress classes extend _UQAddress.

***See Also***

Chapter 13, "Understanding JSMCAPI Classes," Extension Class Constructor, page 348

Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Class Constructor, page 266

Chapter 13, "Understanding JSMCAPI Classes," A2AChatAddress Class Constructor, page 279

Chapter 13, "Understanding JSMCAPI Classes," ChatAddress Class Constructor, page 303

Chapter 13, "Understanding JSMCAPI Classes," EmailAddress Class Constructor, page 332

Chapter 13, "Understanding JSMCAPI Classes," GenericAddress Class Constructor, page 360

## Line

Line class describes the line of the extension for a call task. This version supports one extension with two lines and two extensions with one line in each.

***See Also***

Chapter 13, "Understanding JSMCAPI Classes," Line Class Constructor, page 398

## Connection

Tasks are routed to agents through a connection. Email, chat and generic have a dedicated connection class, like EmailConnection, ChatConnection, and GenericConnection. These connections provide task-specific manipulation functions.

***See Also***

Chapter 13, "Understanding JSMCAPI Classes," ChatConnection Class Constructor, page 308

Chapter 13, "Understanding JSMCAPI Classes," EmailConnection Class Constructor, page 335

## Group

The group class defines the group or queue information. Each session can have one or more group objects.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Group Class Constructor, page 386

## Task

This abstract base class defines a unit of work. The classes that extend task per media type are:

- Call
- Email
- Chat
- A2AChat
- GenericTask

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Call Class Constructor, page 292

Chapter 13, "Understanding JSMCAPI Classes," Email Class Constructor, page 327

Chapter 13, "Understanding JSMCAPI Classes," Chat Class Constructor, page 298

Chapter 13, "Understanding JSMCAPI Classes," A2AChat Class Constructor, page 275

Chapter 13, "Understanding JSMCAPI Classes," GenericTask Class Constructor, page 376

## _User

_User is a base class. The subclasses are User and Buddy. These classes define the properties that pertain to the user such as the addresses, languages, or presence. _User is a virtual class and should not be instantiated.

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," _User Class Constructor, page 271

## MediaType

This class defines the media that an agent can handle.

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," MediaType Class Constructor, page 434

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Reason

The Reason class defines the message or error message that accompanies an event or request. Globalization of the messages is implemented. Furthermore, extra data can be passed in this object for providing a detailed message.

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Reason Class Constructor, page 442

## Statistics

Statistics are provided by the routing server for agent, call, task, group, and user. The following classes describe the statistics for each component:

- AgentStatistics
- CallStatistics
- TaskStatistics
- GroupStatistics
- GroupStatistics1
- GroupStatistics2
- UserStatistics1
- UserStatistics2

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," AgentStatistics Class Constructor, page 283

Chapter 13, "Understanding JSMCAPI Classes," CallStatistics Class Constructor, page 296

Chapter 13, "Understanding JSMCAPI Classes," TaskStatistics Class Constructor, page 467

Chapter 13, "Understanding JSMCAPI Classes," GroupStatistics Constructor, page 390

Chapter 13, "Understanding JSMCAPI Classes," GroupStatistics1 Class Constructor, page 392

Chapter 13, "Understanding JSMCAPI Classes," GroupStatistics2 Class Constructor, page 396

Chapter 13, "Understanding JSMCAPI Classes," UserStatistics1 Class Constructor, page 494

Chapter 13, "Understanding JSMCAPI Classes," UserStatistics2 Class Constructor, page 498

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

## Data

When a task is introduced to the system, data pertaining to the task is defined in the following classes:

- CallData
- EmailData
- ChatData
- GenericData

Application-specific data is provided for tasks via the AppData class. Similarly, user data is defined in UserData.

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," CallData Class Constructor, page 294

Chapter 13, "Understanding JSMCAPI Classes," EmailData Class Constructor, page 346

Chapter 13, "Understanding JSMCAPI Classes," ChatData Class Constructor, page 326

Chapter 13, "Understanding JSMCAPI Classes," GenericData Class Constructor, page 375

## Globals

This class defines functions that are used universally.

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," GLOBALS Class Fields, page 381

## MCEvent

Events passed to the application handler are defined by MCEvent.

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," MCEvent Class Constructor, page 432

## Caps

Capabilities (Caps) define the ability or allowed actions for a component. The following classes define specific capabilities:

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- ChatConnectionCaps
- EmailConnectionCaps
- GenericConnectionCaps
- ExtensionCaps
- LineCaps
- ChatConnectionCaps
- EmaiConnectionCaps
- GenericConnectionCaps
- UserCaps

***See Also***

Chapter 13, "Understanding JSMCAPI Classes," ChatConnectionCaps Class Constructor, page 323

Chapter 13, "Understanding JSMCAPI Classes," EmailConnectionCaps Class Constructor, page 345

Chapter 13, "Understanding JSMCAPI Classes," GenericConnectionCaps Class Constructor, page 373

Chapter 13, "Understanding JSMCAPI Classes," ExtensionCaps Class Constructor, page 356

Chapter 13, "Understanding JSMCAPI Classes," LineCaps Class Constructor, page 427

## ForwardMode

ForwardMode describes various forward modes that can be used while setting the forwarding mode for an Address.

***See Also***

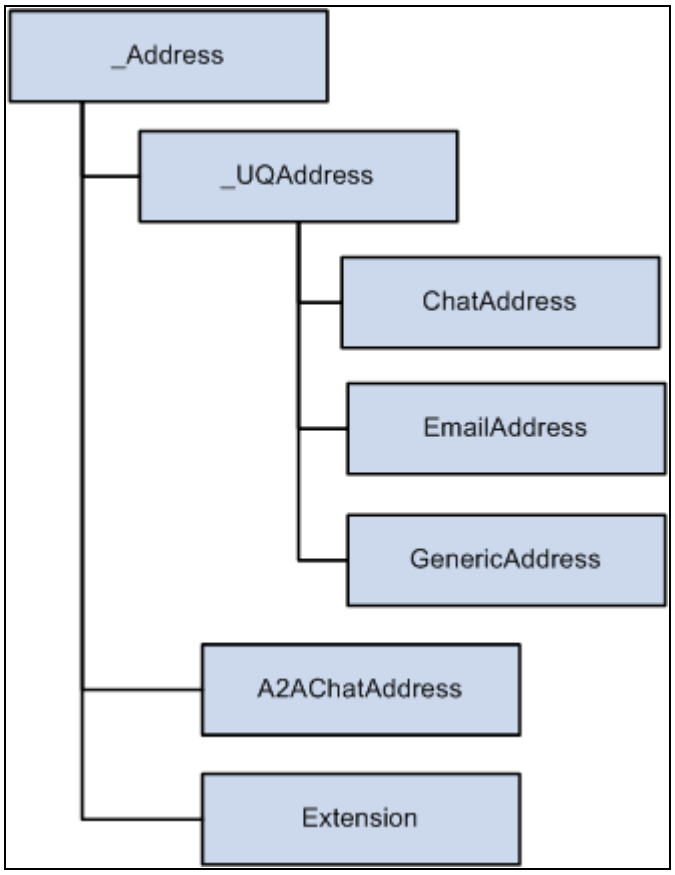
Chapter 13, "Understanding JSMCAPI Classes," ForwardMode Class Constructor, page 359

# _Address Class Hierarchy

The _Address class can be extended by other subclasses. The following flow chart shows the different subclasses and how they interrelate.

Do not instantiate _Address or _UQAddress class objects. Instead, use the child classes.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 




_Address class hierarchy

***See Also***

Chapter 13, "Understanding JSMCAPI Classes," _Address Class Constructor, page 264

Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Class Constructor, page 266

Chapter 13, "Understanding JSMCAPI Classes," A2AChatAddress Class Constructor, page 279

Chapter 13, "Understanding JSMCAPI Classes," Extension Class Constructor, page 348

Chapter 13, "Understanding JSMCAPI Classes," ChatAddress Class Constructor, page 303

Chapter 13, "Understanding JSMCAPI Classes," EmailAddress Class Constructor, page 332

Chapter 13, "Understanding JSMCAPI Classes," GenericAddress Class Constructor, page 360

## _Address Class Constructor

The following is the _Address class constructor.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## _Address

### Syntax

```
_Address()
```

### Description

Creates an _Address object that describes the address.

### Parameters

None.

### Returns

An _Address object.

# _Address Class Fields

This section discusses the JSMCAPI _Address class fields, which are described in alphabetical order.

## caps

### Description

The capacities of the address.

Type: object.

## id

### Description

The ID of the address.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# _Address Class Callback Event Method

The following is the callback event method used with a JSMCAPI _Address object.

## onError

### Syntax

**onError***(event)*

### Description

Fires on the event of an address error.

# _UQAddress Class Constructor

The following is the UQAddress class constructor.

## _UQAddress

### Syntax

**_UQAddress**()

### Description

Creates a _UQAddress object, which is the base for the various addresses associated with the universal queue server.

### Parameters

None.

### Returns

A universal queue address object.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# _UQAddress Class Fields

The _UQAddress class inherits the following fields from the _Address class:

- caps
- id

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Fields, page 265.

The following are the _UQAddress class fields.

## Tasks

### Description

The associative array of the tasks on the queue managed by the address.

# _UQAddress Methods

The following are the _UQAddress methods.

## acceptTask

### Syntax

**acceptTask***(task,reason)*

### Description

Sends a request to the universal queue server to signal that the client has accepted the task.

### Parameters

| ***Parameter*** | ***Description*** |
|---|---|
| *task* | The task ID. |
| *reason* | The associated reason code. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Request number.

## dequeueTask

### Syntax

**dequeueTask**(*task*)

### Description

Sends a request to the universal queue server to remove the task from its queue.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *task* | The task ID. |

### Returns

Request number.

# _UQAddress Class Callback Event Methods

The _UQAddress class inherits the onError callback event method from the _Address class.

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Callback Event Method, page 266.

The following are the callback event methods used with a JSMCAPI universal queue address object. The callback event methods are described in alphabetical order.

## onAccepted

### Syntax

**onAccepted**(*event*)

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when the task is accepted.

## onAcceptingTask

### Syntax

**onAcceptingTask***(event)*

### Description

Fires as the task is being accepted.

## onDequeueingTask

### Syntax

**onDequeueingTask***(event)*

### Description

Fires as the task is being dequeued.

## onNotify

### Syntax

**onNotify***(event)*

### Description

Fires on task notification.

## onTaskAdded

### Syntax

**onTaskAdded***(event)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

### Description

Fires when the task is added to the addressed queue.

## onTaskRemoved

### Syntax

**onTaskRemoved***(event)*

### Description

Fired when the task is removed

## onUnassigned

### Syntax

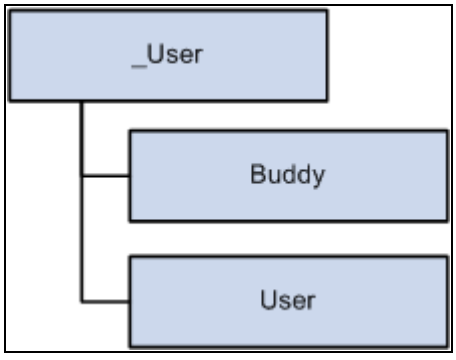
**onUnassigned***(event)*

### Description

Fired when the task is unassigned.

# _User Class Hierarchy

The _User class can be extended by other subclasses. The following flow chart shows the different subclasses and how they interrelate:




_User class hierarchy


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Buddy Class Constructor, page 290

Chapter 13, "Understanding JSMCAPI Classes," User Class Constructor, page 469

# _User Class Constructor

The following is the _User class constructor.

## _User

### Syntax

```
_User()
```

### Description

Describes the base class of user/agent.

### Parameters

None.

### Returns

A _User object.

# _User Class Fields

The following are the fields associated with the JSMCAPI _User class. These fields are discussed in alphabetical order.

## agentID

### Description

The agent's agent ID.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

Type: string.

## caps

### Description

The agent capabilities on each group.

Type: associative array.

## id

### Description

The agent's PeopleSoft user ID.

Type: string.

## name

### Description

The agent's user name.

Type: string.

## presences

### Description

The agent's presence on each group.

Type: associative array.

## states

### Description

Agent state on each group.

Type: associative array, including the constants beginning with ST_*.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

These constants must be accessed from an instantiated object.

## ST_LOGGEDIN

### Description

The agent is logged in.

## ST_LOGGEDOUT

### Description

The agent is logged out.

## ST_NOTREADY

### Description

The agent is not ready.

## ST_READY

### Description

The agent is ready.

## ST_UNKNOWN

### Description

The agent's state is unknown.

## ST_WORKNOTREADY

### Description

The agent is in the work not ready state.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## ST_WORKREADY

### Description

The agent is in the work ready state.

## statistics

### Description

Agent statistics for CTI.

Type: AgentStatistics object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," AgentStatistics Class Constructor, page 283

## statistics1

### Description

Agent statistics for the universal queue server.

Type: UserStatistics1 object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," UserStatistics1 Class Constructor, page 494

## statistics2

### Description

Agent statistics for the universal queue server.

Type: UserStatistcs2 object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," UserStatistics2 Class Constructor, page 498

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# A2AChat Class Constructor

The following is the A2AChat class constructor.

## A2AChat

### Syntax

**A2AChat***(event,address,chatType)*

### Description

Creates an agent-to-agent chat object.

The chat is related to the A2AChatAddress. It does not extend Task as it is not generated or tracked by the universal queue server. Construction occurs inside the A2AChatAddress.

### Parameters

| ***Parameter*** | ***Description*** |
|---|---|
| *event* | Enter the event associated with the agent-to-agent chat. |
| *address* | Enter the address from A2AChatAddress. |
| *chatType* | Enter the chat type, for example consult or answer. |

### Returns

An agent-to-agent chat object.

# A2AChat Class Fields

The following are the A2AChat fields.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## address

### Description

The address containing the agent-to-agent chat.

Type: A2AChatAddress object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," A2AChatAddress Class Constructor, page 279

## agentID

### Description

The associated agent's ID.

Type: string.

## agentName

### Description

The associated agent's name.

Type: string.

## appData

### Description

The application data that is provided to the client with the notification event.

Type: AppData object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," AppData Class Constructor, page 286

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## chatType

### Description

The type of chat, either A2AChat.TYPE_CONSULT or A2AChat.TYPE_ANSWER.

Type: string, with one of the following constants:

| *Value* | *Description* |
|---|---|
| TYPE_ANSWER | The A2AChat type answer. This type is generated when another agent wants this agent to answer an A2AChat. |
| TYPE_CONSULT | The A2AChat type consult. This type is generated when this agent wants to consult another agent. |

### Example

The A2AChat.TYPE_* can only be accessed as in the following example:

```
var value = A2AChat.TYPE_CONSULT;
```

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," GLOBALS Class Fields, page 381

## customerName

### Description

The name of the customer in the ongoing chat.

Type: object.

## id

### Description

The ID of this chat.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

## isConference

### Description

A flag to check whether it is conference or not.

Type: object.

## jr

### Description

JournalRoute

Type: string.

## question

### Description

Question that is asked when customer initiates chat.

Type: object.

## subject

### Description

Subject of the chat. Customer will provide this subject when he initiates chat.

Type: object.

## type

### Description

The task type, chat.

Type: string.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### uniqueueId

#### Description

The unique ID.

Type: string.

## A2AChat Class Method

The following is the A2AChat class method.

### getURL

#### Syntax

**getURL***(defaultURL)*

#### Description

Returns the URL for the given task.

#### Parameters

| *Parameter* | *Description* |
|---|---|
| *defaultURL* | If not null, this value will override the generated base URL. |

#### Returns

The URL associated with this task.

## A2AChatAddress Class Constructor

The following is the A2AChatAddress constructor.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## A2AChatAddress

### Syntax

```
A2AChatAddress()
```

### Description

Agent-to-agent chat address. Handles the creation of agent-to-agent chats for incoming and outgoing chat communication.

### Parameters

None.

### Returns

An A2AChatAddress object.

# A2AChatAddress Class Fields

The A2AChatAddress class inherits the following fields from the _Address class:

- id
- caps

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Fields, page 265.

The following are the fields associated with the A2AChatAddress class.

## id

### Description

The address ID.

Type: string.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## tasks

### Description

The array of A2AChats associated with this address.

Type: associative array.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," GLOBALS Class Fields, page 381

# A2AChatAddress Class Method

The following is the A2AChatAddress class method.

## initiateChat

### Syntax

**initiateChat***(agentId)*

### Description

Initializes the A2AChatAddress by creating an A2AChat task.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *agentId* | The ID of the agent with whom you want to chat. |

### Returns

Request number.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# A2AChatAddress Class Callback Event Methods

The A2AChatAddress class inherits the onError callback event method from the _Address class.

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Callback Event Method, page 266.

The following are the A2AChatAddress class callback methods:

## onChatEnded

### Syntax

**onChatEnded***(event)*

### Description

This event gets called when an agent to agent chat is closed.

## onInitiatingChat

### Syntax

**onInitiatingChat***(event)*

### Description

This event gets called when chat is getting initiated.

## onNotify

### Syntax

**onNotify***(event)*

### Description

This event gets called when there is a new A2Achat notification.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# AgentStatistics Class Constructor

The following is the AgentStatistics class constructor.

## AgentStatistics

### Syntax

```
AgentStatistics()
```

### Description

The agent statistics information.

### Parameters

None.

### Returns

An AgentStatistics object.

# AgentStatistics Class Fields

The following are the AgentStatistics class fields.

## averageCallDuration

### Description

The average call duration, in seconds, for an agent.

Type: number

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## averageHoldDuration

### Description

The average hold duration, in seconds, for an agent.

Type: number.

## callsHandled

### Description

The total number of calls handled by an agent.

Type: number.

## data

### Description

An associative array of key-value pairs that includes all agent statistics.

Type: associative array.

## percentIdleTime

### Description

The percentage of time for which the agent is idle.

Type: number

## percentTimeAvailable

### Description

The percentage of time for which the agent is in ready state.

Type: number


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## percentTimeInCurrentState

### Description

The percentage of time in current state.

Type: number

## percentTimeUnavailable

### Description

The percentage of time for which the agent is in not ready state.

Type: number

## timeCurrentLogin

### Description

The time since login for the agent.

Type: number.

## timeWorking

### Description

The time spent working on tasks.

Type: number.

## totalTaskAcceptedLogin

### Description

The number of tasks accepted since login time

Type: number

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## totalTaskDoneLogin

### Description

Number of tasks done by the agent since login time i.e. current login.

Type: number.

## totalTaskUnassignedLogin

### Description

Number of tasks unassigned since login time

Type: number.

## unavailableDuration

### Description

The total time for which an agent is unavailable.

Type: string.

## waitDuration

### Description

The total time an agent has to wait for a call.

Type: string.

# AppData Class Constructor

The following is the AppData class constructor.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## AppData

### Syntax

```
AppData()
```

### Description

The AppData object describes the key-value pairs of the data received on the agent-to-agent chat event.

### Parameters

None.

### Returns

Type: AppData object.

# AppData Class Fields

The following are the AppData class fields.

## data

### Description

An associative array of the data in name value pairs.

Type: associative array.

## groupId

### Description

The group ID passed in by the notify event.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## jr

### Description

The JournalRoute.

Type: string.

## question

### Description

The initial question of the task.

Type: string.

## strData

### Description

The raw application data as a string.

Type: string.

## subject

### Description

The subject of the task.

Type: string.

## uniqueId

### Description

The unique ID.

Type: string.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## url

### Description

The URL passed in by the AppData.

Type: string with the following constant:

| *Value* | *Description* |
|---|---|
| URL | The associated URL. |

## userId

### Description

The agent's user ID.

Type: string.

## username

### Description

The agent's user name.

Type: string.

## wizUrl

### Description

The wizard URL that is used for popping up new task window.

Type: string.

# AppData Class Method

The following is the AppData class method.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## addKeyValue

### Syntax

**addKeyValue***(key,value)*

### Description

Add the key-value pair to the AppData object.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *key* | The key. |
| *value* | The value. |

### Returns

None.

# Buddy Class Constructor

The Buddy class extends the _User class.

See Chapter 13, "Understanding JSMCAPI Classes," _User Class Constructor, page 271.

The following is the Buddy class constructor.

## Buddy

### Syntax

**Buddy**()

### Description

Describes the buddy. An agent can register buddies and is notified of state changes.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

None.

### Returns

A Buddy object.

## Buddy Class Fields

The Buddy class inherits the following fields from the _User class:

- agentID
- caps
- id
- name
- presences
- states
- statistics
- statistics1
- statistics2

See Chapter 13, "Understanding JSMCAPI Classes," _User Class Constructor, page 271.

## Buddy Class Callback Event Methods

The following are the Buddy class callback event methods.

### onStat1

#### Syntax

**onStat1***(event)*

#### Description

Fires when statistics1 is received.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onStat2

### Syntax

**onStat2***(event)*

### Description

Fires when statistics2 is received.

## onState

### Syntax

**onState***(event)*

### Description

Fires when the state event is received.

# Call Class Constructor

The Call class extends the Task class.

See Chapter 13, "Understanding JSMCAPI Classes," Task Class Constructor, page 464.

The following is the Call class constructor.

## Call

### Syntax

**Call***(strCall)*

### Description

The Call object describes the call task information associated with the line.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *strCall* | Enter the call to use with this object. |

### Returns

A Call object.

## Call Class Fields

The Call class inherits the following fields from the Task class:

- caseid
- cost
- customerid
- group
- id
- onStat
- priority
- type
- urlAbs
- urlRel

See Chapter 13, "Understanding JSMCAPI Classes," Task Class Fields, page 465.

The following are additional Call class fields.

## line

### Description

The line object that is associated with this call.

Type: line object.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Line Class Constructor, page 398

## statistics

### Description

Call statistics object associated with the call.

Type: CallStatistics object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," CallStatistics Class Constructor, page 296

# CallData Class Constructor

The following is the CallData class constructor.

## CallData

### Syntax

```
CallData()
```

### Description

The CallData object describes the key-value pairs of the call data with the call object.

### Parameters

None.

### Returns

A CallData object.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# CallData Class Fields

The following are the CallData class fields:

## ani

### Description

The ANI caller id.

Type: string.

## callId

### Description

The call ID.

Type: string.

## callType

### Description

The call type.

Type: string.

## data

### Description

The key-value pairs of the call data.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### dnis

#### Description

The DNIS callee ID.

Type: string.

## CallData Class Method

The following is the CallData class method.

### addKeyValue

#### Syntax

**addKeyValue***(key),value*

#### Description

Add key value to the CallData object.

#### Parameters

| *Parameter* | *Description* |
|---|---|
| *key* | The key. |
| *value* | The value. |

#### Returns

None.

## CallStatistics Class Constructor

The following is the CallStatistics class constructor.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## CallStatistics

### Syntax

```
CallStatistics()
```

### Description

The call statistics information.

### Parameters

None.

### Returns

A CallStatistics object.

# CallStatistics Class Fields

The following are the CallStatistics class fields.

## data

### Description

An associative array of key-value pairs that includes all statistics.

Type: associative array.

## holdTime

### Description

Duration that the call is on hold.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## queueTime

### Description

Duration that this call has been in the queue.

Type: string.

## talkTime

### Description

Duration that the call is established.

Type: string.

# Chat Class Constructor

The Chat class extends the Task class.

See Chapter 13, "Understanding JSMCAPI Classes," Task Class Constructor, page 464.

The following is the Chat class constructor.

## Chat

### Syntax

**Chat***(event,address)*

### Description

The Chat object describes the task information for tasks associated with the ChatAddress.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *event* | The onNotify event. |
| *address* | The chat address. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

A Chat object.

## Chat Class Fields

The Chat class inherits the following fields from the Task class:

- caseid
- cost
- customerid
- group
- id
- onStat
- priority
- type

  The task type for chat that is Task.TYPE_CHAT.
- urlAbs
- urlRel

See Chapter 13, "Understanding JSMCAPI Classes," Task Class Fields, page 465.

The following are the additional Chat class fields.

## address

### Description

The address containing the agent-to-agent chat.

Type: A2AChatAddress object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," A2AChatAddress Class Constructor, page 279

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## agentId

### Description

The agent's ID.

Type: string.

## appData

### Description

The application data that is provided to the client with the notification event.

Type: AppData object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," AppData Class Constructor, page 286

## chatconnection

### Description

The chat connection object associated with this call.

Type: line object

## chatType

### Description

Type of chat, for example consulting chat or answering chat

Type: string

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## customerName

### Description

The customer username.

Type: string.

## groupId

### Description

The group ID.

Type: string.

## question

### Description

The question for this chat.

Type: string.

## subject

### Description

The subject of the chat.

Type: string.

## statistics

### Description

The chat statistics object that associated with this chat.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## userData

### Description

The user data associated with the chat.

Type: object.

# Chat Class Method

The Chat class extends the Task class.

See Chapter 13, "Understanding JSMCAPI Classes," Task Class Constructor, page 464.

The following is the Chat class method.

## gettpUrl

### Syntax

**gettpUrl***(defaultUrl)*

### Description

Returns the URL for the given task for a third-party routing server.

### Parameters

| *Parameter* | *Description* |
|---|---|
| defaultUrl | If not null, this value will override the generated base URL. |

### Returns

Returns the URL associated with this task.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## getUrl

### Syntax

**getUrl**(*defaultUrl*)

### Description

Returns the URL for the given task.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *defaultUrl* | The default URL for this chat. If not null this value will override the generated base URL. |

### Returns

Returns the URL for the chat.

# ChatAddress Class Constructor

The ChatAddress class extends the _UQAddress class.

See Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Class Constructor, page 266.

The following is the ChatAddress class constructor.

## ChatAddress

### Syntax

**ChatAddress**()

### Description

Handles the creation of customer chat tasks.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

None.

### Returns

A ChatAddress object.

## ChatAddress Class Fields

The ChatAddress class inherits the following fields from the _Address class:

- id

  The address's ID is equal to Task.TYPE_CHAT.

- caps

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Fields, page 265.

The ChatAddress class inherits the following field from the _UQAddress class:

tasks

See Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Class Fields, page 267.

The following are ChatAddress class fields.

### chatconnections

#### Description

List of ChatConnection objects that are associated with chat address.

Type: chatconnection object.

## ChatAddress Class Methods

The ChatAddress class inherits the following methods from the _UQAddress class:

- acceptTask
- dequeueTask

See Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Methods, page 267.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

The following are ChatAddress class methods.

## chat

### Syntax

**chat***(agentId,buddyId,reason)*

### Description

Sends a request to initiate a new chat.

### Parameters

| *Parameter* | *Description* |
|---|---|
| agentId | The source agent ID. |
| buddyId | The Id of the buddy. |
| reason | The reason code for chat request. |

### Returns

Request number.

## getChatconnectionByConnectionId

### Syntax

**getChatconnectionByConnectionId***(connectionId)*

### Description

Get the chatconnection object with the chatconnection Id.

### Parameters

| *Parameter* | *Description* |
|---|---|
| connectionId | The connection id associated with a chat. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Line object, null if there is no connection with that chatconnection id.

## getChatconnectionindexByConnectionId

### Syntax

**`getChatconnectionindexByConnectionId`**
*`(connectionId)`*

### Description

Get the chatconnection index with the chatconnection Id.

### Parameters

| *Parameter* | *Description* |
|---|---|
| connectionId | The connection id associated with a chat. |

### Returns

index, −1 if there is no connection with that chatconnection id.

## getFreeChatconnection

### Syntax

**`getFreeChatconnection`**`()`

### Description

Get the free chatconnection object. A connection is free if there is no activity on this line.

### Returns

Returns chatconnection object, null if there is no free line.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## getFreeChatconnectionIndex

### Syntax

**`getFreeChatconnectionIndex`**`()`

### Description

Get the free chatconnection index. A connection is free if there is no activity on this line.

### Returns

Returns chatconnection index, null if there is no free line.

# ChatAddress Callback Event Methods

The ChatAddress class inherits the onError callback event method from the _Address class.

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Callback Event Method, page 266.

The ChatAddress class inherits the following callback event methods from the _UQAddress class:

- onAccepted
- onAcceptingTask
- onDequeueingTask
- onNotify
- onTaskAdded
- onTaskRemoved
- onUnassigned

See Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Class Fields, page 267.

The following are ChatAddress callback event methods:

## onCapabilitiesChanged

### Syntax

**`onCapabilitiesChanged`***`event`*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when ChatAddress capabilities change

## ChatConnection Class Constructor

The following is the chatconnection class constructor.

### ChatConnection

#### Syntax

```
ChatConnection()
```

#### Description

Chat connection object.

#### Parameters

None

#### Returns

Returns a chatconnection object.

## ChatConnection Class Fields

The following are chatconnection class fields:

### caps

#### Description

The capabilities of the ChatConnection.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## chat

### Description

Chat task associated with the connection.

Type: string

## connectionId

### Description

The connection Id.

Type: string

## id

### Description

The constant representing ChatConnection Id.

Type: string

## state

### Description

The connection state.

Type: string

| *Value* | *Description* |
|---|---|
| ST_DIALING | Dialing state. |
| ST_DROPPED | Dropped state. |
| ST_IDLE | Idle state. |
| ST_INCOMING | Incoming state. |
| ST_TALKING | Talking state. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Value* | *Description* |
|---|---|
| ST_WRAPUP | Wrap up state. |

# ChatConnection Class Methods

The following are chatconnection class methods:

## answer

### Syntax

**answer***(agentId,reason)*

### Description

Answer a chat request.

### Parameters

| *Parameter* | *Description* |
|---|---|
| agent Id | The Id of the agent answering the call. |
| reason | The reason code for the answer request. |

### Returns

Request number.

## attachUserData

### Syntax

**attachUserData***(userData,reason)*

### Description

Attach user data for the chat connection.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| userData | User data of the chat connection. |
| reason | The reason code for the answer request. |

### Returns

Request number.

## conference

### Syntax

**conference***(agentId,reason)*

### Description

Conference a chat with another agent.

### Parameters

| *Parameter* | *Description* |
|---|---|
| agent Id | The Id of the agent invited to the conference. |
| reason | The reason code for the conference request. |

### Returns

Request number.

## forward

### Syntax

**forward***(fromagentId,toagentId,qid,userdata,chatdata,reason)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Forward chat to another agent or another queue.

### Parameters

| *Parameter* | *Description* |
|---|---|
| fromagentid | The agent id from whom task is forwarded. |
| toagentid | The target agentid to whom task is forwarded. |
| qid | The target queue to forward the task. |
| userdata | The user data. |
| chatdata | The chat data. |
| Reason | The reason code to forward the task. |

### Returns

Request number.

## gethistory

### Syntax

**`gethistory`***`(agentId,noofLines,reason)`*

### Description

Request chat history by specifying number of lines of chat history needed.

### Parameters

| *Parameter* | *Description* |
|---|---|
| agent Id | The source agent ID. |
| noofLines | Number of lines of history required. |
| reason | The reason code for the history request. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Request number.

## getUrl

### Syntax

**getUrl***(defaultURL)*

### Description

Get the URL for screen popup.

### Parameters

| *Parameter* | *Description* |
|---|---|
| defaultUrl | The default popup URL. |

### Returns

Returns a URL string.

## message

### Syntax

**message***(agentId,message,reason)*

### Description

Send a chat message while chatting.

### Parameters

| *Parameter* | *Description* |
|---|---|
| agent Id | The source agent ID. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Parameter* | *Description* |
|---|---|
| message | The message to send to other party. |
| reason | The reason code for the message request. |

### Returns

Request number.

## pushURL

### Syntax

**pushURL***(URL)*

### Description

Request to push a URL to the customer.

### Parameters

| *Parameter* | *Description* |
|---|---|
| URL | URL to push. |

### Returns

Request number.

## reject

### Syntax

**reject***(agentId,reason)*

### Description

Reject a chat.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| agent Id | The source agent ID. |
| reason | The reason code for reject. |

### Returns

Request number.

## release

### Syntax

**release**(*agentId,reason)*

### Description

Release a chat.

### Parameters

| *Parameter* | *Description* |
|---|---|
| agent Id | The source agent ID. |
| reason | The reason code for release request. |

### Returns

Request number.

## typing

### Syntax

**typing**(*agentId,reason)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Indicates typing in a chat conversation by one party.

### Parameters

| *Parameter* | *Description* |
|---|---|
| agent Id | The source agent ID. |
| reason | The reason code for typing. |

### Returns

Request number.

## wrapup

### Syntax

**wrapup***(agentid,message,tasknum,reason)*

### Description

Stores chat wrap up comments.

### Parameters

| *Parameter* | *Description* |
|---|---|
| agentid | The agent id who enters wrap up comments. |
| message | Chat message. |
| tasknum | Task number. |
| reason | The reason code. |

### Returns

Request number.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# ChatConnection Class Callback Event Methods

The following are ChatConnection class callback event methods.

## onAccepted

### Syntax

**onAccepted***(event)*

### Description

Fires when chat is accepted by a agent.

## onAnswering

### Syntax

**onAnswering***(event)*

### Description

Fires when answering the chat.

## onCapabilitiesChanged

### Syntax

**onCapabilitiesChanged***(event)*

### Description

Fires when chat capabilities change.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onChatdataChanged

### Syntax

**onChatdataChanged***(event)*

### Description

Fires when there is a change in chat data.

## onConferencing

### Syntax

**onConferencing***(event)*

### Description

Fires when conferencing the chat.

## onDialing

### Syntax

**onDialing***(event)*

### Description

Fires when chat is in the process of connecting.

## onDropped

### Syntax

**onDropped***(event)*

### Description

Fires when chat is dropped.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onError

### Syntax

**onError***(event)*

### Description

Fires when a there is a chat connection error.

## onForwarded

### Syntax

**onForwarded***(event)*

### Description

Fires when chat is forwarded.

## onForwardError

### Syntax

**onForwardError***(event)*

### Description

Fires when there is error in forwarding the task.

## onForwarding

### Syntax

**onForwarding***(event)*

### Description

Fires when forwarding the chat.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onHistory

### Syntax

**onHistory***(event)*

### Description

Fires when agent receives chat history.

## onIncomingChat

### Syntax

**onIncomingChat***(event)*

### Description

Fires when ChatConnection state is changed to INCOMING.

## onMessage

### Syntax

**onMessage***(event)*

### Description

Fires when there is incoming message.

## onPartyAdded

### Syntax

**onPartyAdded***(event)*

### Description

Fires when a new chat party is added.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onPartyChanged

### Syntax

**onPartyChanged***(event)*

### Description

Fires when the chat party changes.

## onPartyRemoved

### Syntax

**onPartyRemoved***(event)*

### Description

Fires when a chat party is removed.

## onProperties

### Syntax

**onProperties***(event)*

### Description

Fires when a agent receives chat properties.

## onPushURL

### Syntax

**onPushURL***(event)*

### Description

Fires when a routing system push URL.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onRejected

### Syntax

**onRejected***(event)*

### Description

Fires when a chat is rejected.

## onReleased

### Syntax

**onReleased***(event)*

### Description

Fires when a chat is released.

## onReleasing

### Syntax

**onReleasing***(event)*

### Description

Fires when conferencing the chat.

## onRevoked

### Syntax

**onRevoked***(event)*

### Description

Fires when a chat is revoked.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onTalking

### Syntax

**onTalking***(event)*

### Description

Fires when an agent/customer is in the state of talking.

## onTyping

### Syntax

**onTyping***(event)*

### Description

Fires when typing in a chat conversation.

## onUserdataChanged

### Syntax

**onUserdataChanged***(event)*

### Description

Fires when there is a change in user data.

# ChatConnectionCaps Class Constructor

The following is the chatconnectioncaps class constructor.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## ChatConnectionCaps

### Syntax

**ChatConnectionCaps***(strCaps)*

### Description

Describes the ChatConnection capabilities.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *strCaps* | Chat connection capability. |

### Returns

Returns a chatconnection object.

# ChatConnectionCaps Class Fields

The following are the ChatConnectionCaps class fields.

## canAnswer

### Description

Answer capability.

Type: boolean

## canConference

### Description

Conference capability.

Type: boolean


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## canConferenceSingle

### Description

Conference single capability.

Type: boolean

## canForward

### Description

Forward capability.

Type: boolean

## canIndicateTyping

### Description

Indicate Typing capability.

Type: boolean

## canPushURL

### Description

PushURL capability.

Type: boolean

## canReject

### Description

Reject capability.

Type: boolean

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### canSendMessage

#### Description

SendMessage capability.

Type: boolean

## ChatData Class Constructor

The following is ChatData class constructor.

### ChatData

#### Syntax

```
ChatData()
```

#### Description

The ChatData object describes the key-value pairs of the chat data with the chat object.

## ChatData Class Fields

The following is ChatData class field.

### data

#### Description

Key value pairs that includes data.

Type: object.

## ChatData Class Methods

The following is ChatData class method.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## addKeyValue

### Syntax

**addKeyValue***(key,value )*

### Description

Add key value to the chatdata object.

### Parameters

| *Parameter* | *Description* |
|---|---|
| key | The variable name. |
| value | The value of the variable. |

# Email Class Constructor

The Email class extends the Task class.

See Chapter 13, "Understanding JSMCAPI Classes," Task Class Constructor, page 464.

The following is the email class constructor.

## Email

### Syntax

**Email***(event)*

### Description

The Email object describes the task information for tasks associated with the EmailAddress.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *event* | The notification event. |

### Returns

Returns an Email object.

## Email Class Fields

The Chat class inherits the following fields from Task class:

- caseid
- cost
- customerid
- group
- id
- onStat
- priority
- type

  The task type is equal to Task.TYPE_EMAIL.
- urlAbs
- urlRel

See Chapter 13, "Understanding JSMCAPI Classes," Task Class Fields, page 465.

The following are the additional Email class fields.

## address

### Description

The address containing the A2AChat.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## agentId

### Description

The agent's ID.

## appData

### Description

The application data that is provided to the client with the notification event.

Type: AppData object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Task Class Fields, page 465

## customerName

### Description

The customer username.

Type: string.

## emailconnection

### Description

The emailconnection object that associated with this email.

Type: line object.

## emailId

### Description

The unique email id generated from an Enqueue. It identifies the email stored in the database ie. it serves as a key to the data stored in the database.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Type: string.

## groupId

### Description

The group id.

Type: string.

## question

### Description

The question for this email.

Type: string.

## statistics

### Description

The statistics object that is associated with this email.

Type: object.

## subject

### Description

The subject of this email.

Type: string.

## userData

### Description

The user data that is associated with this email.

Type: object.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Email Class Method

The following are the Email class methods:

## gettpUrl

### Syntax

**gettpUrl**(*defaultUrl*)

### Description

Returns the URL for the given task for third-party routing.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *defaultUrl* | If not null, this value will override the generated base URL. |

### Returns

The URL associated with this task.

## getUrl

### Syntax

**getURL**(*defaultUrl*)

### Description

Returns the URL for the given task.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *defaultUrl* | If not null, this value will override the generated base URL. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

The URL associated with this task.

## EmailAddress Class Constructor

The EmailAddress class extends the _UQAddress class.

See Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Class Constructor, page 266.

The following is the EmailAddress class constructor.

### EmailAddress

#### Syntax

```
EmailAddress()
```

#### Description

Handles the creation of email tasks.

#### Parameters

None.

#### Returns

An EmailAddress object.

## EmailAddress Class Fields

The EmailAddress class inherits the following fields from the _Address class:

- id

  The address's ID is equal to Task.TYPE_EMAIL.

- caps

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Fields, page 265.

The EmailAddress class inherits the tasks field from the _UQAddress class.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

See Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Class Fields, page 267.

The following are EmailAddress class fields.

## agent

### Description

The agent id of this email address.

## emailconnections

### Description

Describes the EmailConnection object.

# EmailAddress Class Methods

The EmailAddress class inherits the following fields from the _UQAddress class:

- acceptTask
- dequeueTask

See Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Methods, page 267.

## getEmailconnectionByConnectionId

### Syntax

**`getEmailconnectionByConnectionId`***`(connectionId)`*

### Description

Get the emailconnection object with the emailconnection id.

### Parameters

| ***Parameter*** | ***Description*** |
|---|---|
| connectionId | The connection id associated with a email. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Returns line object, null if there is no connection with that emailconnection id.

## getEmailconnectionindexByConnectionId

### Syntax

**`getEmailconnectionindexByConnectionId`***`(connectionId)`*

### Description

Get the emailconnection index with the emailconnection id.

### Parameters

| *Parameter* | *Description* |
|---|---|
| connectionId | The connection id associated with a email. |

### Returns

Returns index, -1 if there is no connection with that emailconnection id.

## getFreeEmailconnection

### Syntax

**`getFreeEmailconnection`**`()`

### Description

Get the free email connection object. A connection is free if there is no activity on this line.

### Returns

Returns emailconnection object, null if there is no free line.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# EmailAddress Callback Event Methods

The EmailAddress class inherits the onError callback event method from the _Address class.

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Callback Event Method, page 266.

The EmailAddress class inherits the following callback event methods from the _UQAddress class:

- onAccepted
- onAcceptingTask
- onDequeueingTask
- onNotify
- onTaskAdded
- onTaskRemoved
- onUnassigned

See Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Class Callback Event Methods, page 268.

# EmailConnection Class Constructor

The following is EmailConnection class constructor.

## EmailConnection

### Syntax

```
EmailConnection()
```

### Description

Handles email connection.

# EmailConnection Class Fields

The following are EmailConnection class fields.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## caps

### Description

The capabilities of the EmailConnection.

## connectionId

### Description

The email connection Id.

Type: string

## email

### Description

Email task id associated with this email connection.

Type: string with following constants:

## id

### Description

The id of the EmailConnection.

Type: string

## state

### Description

The connection state.

Type: string

| *Value* | *Description* |
|---|---|
| ST_DIALING | Dialing state. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Value* | *Description* |
|---|---|
| ST_DROPPED | Dropped state. |
| ST_IDLE | Idle state. |
| ST_INCOMING | Incoming state. |
| ST_TALKING | Talking state. |

# EmailConnection Class Methods

The following are the EmailConnection class methods:

## abandon

### Syntax

**abandon***(reason)*

### Description

Abandon email task without completing the task .

### Parameters

| *Parameter* | *Description* |
|---|---|
| Reason | The reason code to abandon. |

### Returns

Request number.

## answer

### Syntax

**answer***(reason)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Answer/Accept the email assignment.

### Parameters

| *Parameter* | *Description* |
|---|---|
| Reason | The reason code to answer. |

### Returns

Request number.

## attachUserData

### Syntax

**attachUserData***(userData,reason)*

### Description

Attach user data for the email connection.

### Parameters

| *Parameter* | *Description* |
|---|---|
| userData | User data of the email connection. |
| reason | The reason code for the answer request. |

### Returns

Request number.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## complete

### Syntax

**complete***(reason)*

### Description

Send task completion notification.

### Parameters

| *Parameter* | *Description* |
|---|---|
| Reason | The reason code to complete. |

### Returns

Request number.

## forward

### Syntax

**forward***(fromagentId,toagentId,qid,userdata,emaildata,reason)*

### Description

Forward email to another agent or to another queue.

### Parameters

| *Parameter* | *Description* |
|---|---|
| fromagentid | The agent id from whom task is forwarded. |
| toagentid | The target agent id to whom task is forwarded. |
| qid | The target queue to forward the task. |
| userdata | The user data. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Parameter | Description |
|---|---|
| emaildata | The email data. |
| Reason | The reason code to forward the task. |

### Returns

Request number.

## reject

### Syntax

**reject***(reason)*

### Description

Reject email assignment.

### Parameters

| Parameter | Description |
|---|---|
| Reason | The reason code to reject. |

### Returns

Request number.

## withdraw_RES

### Syntax

**withdraw_RES**
*(reason)*

### Description

Send response for the request of (Withdraw of Email) by routing server. This is the response from the agent to the routing system.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| Reason | The reason code to withdraw. |

### Returns

Request number.

## EmailConnection Class Callback Event Methods

The following are EmailConnection class callback event methods:

## onAnswering

### Syntax

**onAnswering***(event)*

### Description

Fires when email task is getting answered.

## onCapabilitiesChanged

### Syntax

**onCapabilitiesChanged***(event)*

### Description

Fires when there is a change in connection capabilities.

## onCompleted

### Syntax

**onCompleted***(event)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when EmailConnection state is changed to COMPLETED.

## onDropped

### Syntax

**onDropped***(event)*

### Description

Fires when EmailConnection state is changed to DROPPED.

## onEmaildataChanged

### Syntax

**onEmaildataChanged***(event)*

### Description

Fires when there is a change in email data.

## onError

### Syntax

**onError***(event)*

### Description

Fires when a there is a email connection error.

## onForwarded

### Syntax

**onForwarded***(event)*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when email is forwarded to another queue or to another agent.

## onForwardError

### Syntax

**onForwardError***(event)*

### Description

Fires when there is error in forwarding the task.

## onForwarding

### Syntax

**onForwarding***(event)*

### Description

Fires when forwarding the email.

## onIncoming

### Syntax

**onIncoming***(event)*

### Description

Fires when EmailConnection state is changed to INCOMING.

## onProcessing

### Syntax

**onProcessing***(event)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when EmailConnection state is changed to PROCESSING.

## onRejected

### Syntax

**onRejected***(event)*

### Description

Fires when email task is rejected by agent and the routing system acknowledges the same.

## onRevoked

### Syntax

**onRevoked***(event)*

### Description

Fires when email is revoked by routing system before agent accepts that.

## onUserdataChanged

### Syntax

**onUserdataChanged***(event)*

### Description

Fires when there is a change in user data.

## onWithdraw_REQ

### Syntax

**onWithdraw_REQ***(event)*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when there is a request for email withdraw from routing server. This is a request from routing server to the agent.

## EmailConnectionCaps Class Constructor

The following is EmailConnectionCaps class constructor:

## EmailConnectionCaps

### Syntax

**`EmailConnectionCaps`***`(strCaps)`*

### Description

Describes the EmailConnection capabilities.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *strCaps* | The capability of the connection |

### Returns

Returns EmailConnection object.

## EmailConnectionCaps Class Fields

The following are the EmailConnectionCaps class fields.

## canAnswer

### Description

Answer/Accept an email capability.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Type: boolean

## canComplete

### Description

Capability to complete an assigned email.

Type: boolean

## canForward

### Description

Capability to forward an email.

Type: boolean

## canReject

### Description

Capability to reject an email.

Type: boolean

# EmailData Class Constructor

The following is EmailConnectionCaps class constructor.

## EmailData

### Syntax

```
EmailData()
```

### Description

The EmailData object describes the key-value pairs of the email data with the Email object.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Returns EmailData object.

## EmailData Class Fields

The following is EmailData class field.

### data

#### Description

Key value pairs that includes email data.

Type: object.

## EmailData Class Methods

The following is EmailData class method.

### addKeyValue

#### Syntax

**addKeyValue***(key,value )*

#### Description

Add key value to the emaildata object.

#### Parameters

| *Parameter* | *Description* |
|---|---|
| key | The variable name. |
| value | The value of the variable. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Extension Class Constructor

The Extension class extends the _Address class.

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Constructor, page 264.

The following is the Extension class constructor.

## Extension

### Syntax

**`Extension`***`(numOfLines)`*

### Description

The Extension object describes the CTI address.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *numOfLines* | Enter the number of lines for this extension. |

### Returns

An Extension object.

# Extension Class Fields

The Extension class inherits the following fields from the _Address class:

- caps
- id

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Fields, page 265.

The following are the Extension class fields.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## forwardMode

### Description

The forward mode of the extension.

Type: forwardMode object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," ForwardMode Class Constructor, page 359

## isDnd

### Description

Flag for Do Not Disturb.

Type: boolean.

## lines

### Description

List of line objects associated with this extension.

Type: list.

## numOfLines

### Description

The number of lines for this extension.

Type: number.

# Extension Class Methods

The Extension class inherits the onError method from the Address class.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

The following are the Extension class methods.

## cancelDnd

### Syntax

**cancelDnd**(*reason*)

### Description

Cancel the DND (do not disturb).

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason for DND cancellation. |

### Returns

Request number.

## cancelForwardSet

### Syntax

**cancelForwardSet**(*reason*)

### Description

Cancel the forward.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason for cancelling the forward. |

### Returns

Request number.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## forwardSet

### Syntax

**forwardSet***(number,mode,reason)*

### Description

Forward the call to another number/extension.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *mode* | The forward mode. |
| *number* | The number to which all calls will be forwarded. |
| *reason* | The reason to forward. |

### Returns

Request number.

## getDialingLine

### Syntax

**getDialingLine**()

### Description

Get the dialing line object.

### Parameters

None

### Returns

Returns a line object. Null if there is no dialing line.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Line Class Constructor, page 398

## getFreeLine

### Syntax

```
getFreeLine()
```

### Description

Get the free line object. A line is free if there is no activity on this line.

### Parameters

None

### Returns

Returns a line object. It returns null if there is no free line.

## getLineByConnectionId

### Syntax

```
getLineByConnectionId(connectionID)
```

### Description

Get the line object with the call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *connectionId* | The connection ID associated with a call. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Returns a Line object. Returns null if there is no line with that call.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Line Class Constructor, page 398

## getOffHookLine

### Syntax

**`getOffHookLine`**`()`

### Description

Get the offhook line object.

### Parameters

None.

### Returns

Returns a Line object. Returns null if there is no offhook line.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Line Class Constructor, page 398

## setDnd

### Syntax

**`setDnd`***`(reason)`*

### Description

Fires when setting DND (do not disturb).

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to set the DND. |

### Returns

Request number.

## Extension Class Callback Event Methods

The following are the Extension class callback event methods.

## onCancelingDnd

### Syntax

**onCancelingDnd***(event)*

### Description

Fires when canceling DND (do not disturb).

## onCancelingForward

### Syntax

**onCancelingForward***(event)*

### Description

Fires when canceling forward.

## onDnd

### Syntax

**onDnd***(event)*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when DND (Do Not Disturb) is requested and processed.

## onDndCanceled

### Syntax

**onDndCanceled***(event)*

### Description

Fires when DND is cancelled.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *event* | The event object. |

### Returns

None.

## onForwardCanceled

### Syntax

**onForwardCanceled***(event)*

### Description

Fires when forward is canceled.

## onForwarded

### Syntax

**onForwarded***(event)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when call is forwarded.

## onForwarding

### Syntax

**onForwarding***(event)*

### Description

Fires when forwarding the call.

## onSettingDnd

### Syntax

**onSettingDnd***(event)*

### Description

Fires when setting DND (do not disturb).

# ExtensionCaps Class Constructor

The following is the Extension class constructor.

## ExtensionCaps

### Syntax

**ExtensionCaps***(strCaps)*

### Description

Describes the extension's capabilities.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *strCaps* | A string comprising the extension's capabilities. |

### Returns

An ExtensionCaps object.

## ExtensionCaps Class Fields

The following are the Extension class fields.

## canCancelDnd

### Description

This extension can cancel DND (do not disturb).

Type: boolean.

## canDial

### Description

This extension can dial out.

Type: boolean.

## canFwdBusy

### Description

This extension can forward calls if busy.

Type: boolean.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## canFwdBusyNoAnswer

### Description

This extension can forward calls if busy/no answer.

Type: boolean.

## canFwdCancelForward

### Description

This extension can cancel forward.

Type: boolean.

## canFwdDefault

### Description

This extension can forward.

Type: boolean.

## canFwdNoAnswer

### Description

This extension can forward if no answer.

Type: boolean.

## canFwdUnconditional

### Description

This extension can forward unconditionally.

Type: boolean.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## canRefreshState

### Description

This extension can refresh state.

Type: boolean.

## canSetDnd

### Description

This extension can set DND (do not disturb).

Type: boolean.

# ForwardMode Class Constructor

The following is the ForwardMode class constructor.

## ForwardMode

### Syntax

```
ForwardMode()
```

### Description

Describes various forward modes that can be used while setting the forwarding mode for an Address.

### Parameters

None.

### Returns

A ForwardMode object.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# ForwardMode Class Field

The following is the ForwardMode class field.

## mode

### Description

The current forwarding mode.

Type: string of the following constants.

| *Value* | *Description* |
|---|---|
| BUSY | The BUSY forwarding mode. |
| BUSYNOANSWER | The BUSYNOANSWER forwarding mode. |
| DEFAULT | The DEFAULT forwarding mode. |
| NOANSWER | The NOANSWER forwarding mode. |
| NONE | No forwarding mode. |
| UNCONDITIONAL | The UNCONDITIONAL forwarding mode. |

# GenericAddress Class Constructor

The GenericAddress class extends the _UQAddress class.

The following is the GenericAddress class constructor.

## GenericAddress

### Syntax

```
GenericAddress()
```

### Description

Handles the creation of generic tasks.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

None.

### Returns

A GenericAddress object.

---

## GenericAddress Class Fields

The GenericAddress class inherits the following fields from the _Address class.

- caps
- id

  The address's ID is equal to Task.TYPE_GENERIC.

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Fields, page 265.

The GenericAddress class inherits the tasks field from the _UQAddress class.

See Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Class Fields, page 267.

The following are GenericAddress class fields:

## agent

### Description

The agent id of this generic address owner.

Type: object

## genericconnections

### Description

List of GenericConnections.

Type: genericconnectionobject

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# GenericAddress Class Methods

The GenericAddress class inherits the following methods from the _UQAddress class:

- acceptTask
- dequeueTask

See Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Methods, page 267.

The following are GenericAddress own class methods.

## getFreeGenericconnection

### Syntax

**getFreeGenericconnectio**()

### Description

Get the free generic connection object. A connection is free if there is no activity on this line.

### Parameters

None.

### Returns

Returns genericconnection object, null if there is no free line.

## getGenericconnectionByConnectionId

### Syntax

**getFreeGenericconnectio***(connectionId)*

### Description

Get the genericconnection object with the genericconnection id.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| connectionId | The connection id associated with a generic. |

### Returns

Returns line object, null if there is no connection with that genericconnection id.

## getGenericconnectionindexByConnectionId

### Syntax

**`getGenericconnectionindexByConnectionId`***`(connectionId)`*

### Description

Get the genericconnection index with the genericconnection id.

### Parameters

| *Parameter* | *Description* |
|---|---|
| connectionId | The connection id associated with a generic. |

### Returns

Returns index, -1 if there is no connection with that genericconnection id.

# GenericAddress Class Callback Event Methods

The GenericAddress class inherits the onError callback event method from the _Address class.

See Chapter 13, "Understanding JSMCAPI Classes," _Address Class Callback Event Method, page 266.

The GenericAddress class inherits the following callback event methods from the _UQAddress class:

- onAcceptingTask
- onDequeueingTask

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- onTaskAdded
- onTaskRemoved
- onNotify
- onAccepted
- onUnassigned

See Chapter 13, "Understanding JSMCAPI Classes," _UQAddress Class Callback Event Methods, page 268.

# GenericConnection Class Constructor

The following is the GenericConnection class constructor.

## GenericConnection

### Syntax

```
GenericConnection()
```

### Description

Handles Generic connection.

### Parameters

None.

### Returns

A genericconnection object.

# GenericConnection Class Fields

The following are GenericConnection class fields:

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## caps

### Description

The capabilities of the GenericConnection.

## connectionId

### Description

The generic connection Id.

Type: string

## generic

### Description

The generic task id associated with this generic connection.

Type: string

## id

### Description

The id of the GenericConnection.

Type: string

## state

### Description

The connection state.

Type: string with following constants:

| *Value* | *Description* |
|---|---|
| ST_DIALING | Dialing state. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Value* | *Description* |
|---|---|
| ST_DROPPED | Dropped state. |
| ST_IDLE | Idle state. |
| ST_INCOMING | Incoming state. |
| ST_TALKING | Talking state. |

# GenericConnection Class Methods

The following are the GenericConnection class methods:

## abandon

### Syntax

**abandon***(reason)*

### Description

Abandon generic task.

### Parameters

| *Parameter* | *Description* |
|---|---|
| Reason | The reason code to abandon. |

### Returns

Request number.

## answer

### Syntax

**answer***(reason)*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Answer/Accept the generic task assignment.

### Parameters

| *Parameter* | *Description* |
|---|---|
| Reason | The reason code to answer. |

### Returns

Request number.

## attachUserData

### Syntax

**attachUserData***(userData,reason)*

### Description

Attach user data for the generic connection.

### Parameters

| *Parameter* | *Description* |
|---|---|
| userData | User data of the generic connection. |
| reason | The reason code for the answer request. |

### Returns

Request number.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## complete

### Syntax

**complete***(reason)*

### Description

Send task completion notification.

### Parameters

| *Parameter* | *Description* |
|---|---|
| Reason | The reason code to complete. |

### Returns

Request number.

## forward

### Syntax

**forward***(fromagentId,toagentId,qid,userdata,genericdata,reason)*

### Description

Forward generic task to another Agent or to another queue.

### Parameters

| *Parameter* | *Description* |
|---|---|
| fromagentid | The agent id from whom task is forwarded. |
| toagentid | The target agent id to whom task is forwarded. |
| qid | The target queue to forward the task. |
| userdata | The user data. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Parameter* | *Description* |
|---|---|
| emaildata | The email data. |
| Reason | The reason code to forward the task. |

### Returns

Request number.

## reject

### Syntax

**reject***(reason)*

### Description

Reject generic task assignment.

### Parameters

| *Parameter* | *Description* |
|---|---|
| Reason | The reason code to reject. |

### Returns

Request number.

## withdraw_RES

### Syntax

**withdraw_RES**
*(reason)*

### Description

Send response for the request of (Withdraw of Generic task) by routing server. This is the response from the agent to the routing system.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| Reason | The reason code to withdraw. |

### Returns

Request number.

## GenericConnection Class Callback Event Methods

The following are ChatConnection class callback event methods:

## onCapabilitiesChanged

### Syntax

**onCapabilitiesChanged***(event)*

### Description

Fires when there is a change in GenericData.

## onCompleted

### Syntax

**onCompleted***(event)*

### Description

Fires when GenericConnection state is changed to COMPLETED.

## onDropped

### Syntax

**onDropped***(event)*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when GenericConnection state is changed to DROPPED.

## onError

### Syntax

**onError***(event)*

### Description

Fires when a there is a generic connection error.

## onForwarded

### Syntax

**onForwarded***(event)*

### Description

Fires when generic task is forwarded.

## onForwardError

### Syntax

**onForwardError***(event)*

### Description

Fires when there is error in forwarding the generic task.

## onForwarding

### Syntax

**onForwarding***(event)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when forwarding the generic task.

## onGenericdataChanged

### Syntax

**onGenericdataChanged***(event)*

### Description

Fires when there is a change in GenericData.

## onIncoming

### Syntax

**onIncoming***(event)*

### Description

Fires when GenericConnection state is changed to INCOMING.

## onProcessing

### Syntax

**onProcessing***(event)*

### Description

Fires when GenericConnection state is changed to PROCESSING.

## onRejected

### Syntax

**onRejected***(event)*

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when generic task is rejected by agent and the routing system acknowledges the same.

## onRevoked

### Syntax

**onRevoked***(event)*

### Description

Fires when generic task is revoked by routing system before agent accepts that.

## onUserdataChanged

### Syntax

**onUserdataChanged***(event)*

### Description

Fires when there is a change in user data.

## onWithdraw_REQ

### Syntax

**onWithdraw_REQ***(event)*

### Description

Fires when there is a request for generic task withdraw from routing server. This is a request from routing server to the agent.

# GenericConnectionCaps Class Constructor

The following is GenericConnectionCaps class constructor.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## GenericConnectionCaps

### Syntax

**GenericConnectionCaps**(*strCaps*)

### Description

Describes the GenericConnection capabilities.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *strCaps* | The capability of the connection. |

### Returns

Returns GenericConnection object.

# GenericConnectionCaps Class Fields

The following are the GenericConnectionCaps class fields

## canAnswer

### Description

Answer/Accept a generic task capability.

Type: boolean

## canComplete

### Description

Complete an assigned generic task capability.

Type: boolean


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### canForward

#### Description

Forward a generic task capability.

Type: boolean

### canReject

#### Description

Reject a generic task capability.

Type: boolean

## GenericData Class Constructor

The following is GenericConnectionCaps class constructor:

### GenericData

#### Syntax

```
GenericData()
```

#### Description

The GenericData object describes the key-value pairs of the generic data with the generic object

#### Returns

Returns GenericData object.

## GenericData Class Fields

The following is GenericData class field:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## data

### Description

Key value pairs that includes data.

Type: object.

# GenericData Class Methods

The following is GenericData class method:

## addKeyValue

### Syntax

**addKeyValue***(key,value )*

### Description

Add key value to the genericdata object.

### Parameters

| *Parameter* | *Description* |
|---|---|
| key | The variable name. |
| value | The value of the variable. |

# GenericTask Class Constructor

The GenericTask class extends the Task class.

The following is the GenericTask class constructor


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## GenericTask

### Syntax

**GenericTask***(event)*

### Description

The GenericTask object describes the generic task information for tasks associated with the GenericAddress object.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *event* | The triggering event. |

### Returns

A GenericTask object.

# GenericTask Class Fields

The GenericTask class inherits the following fields from the Task class:

- caseid
- cost
- customerid
- group
- id
- onStat
- priority
- type

  The task type for generic task that is Task.TYPE_GENERIC.
- urlAbs
- urlRel

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

See Chapter 13, "Understanding JSMCAPI Classes," Task Class Fields, page 465.

The following are the additional GenericTask class fields.

## address

### Description

The address containing the A2AChat.

## agentId

### Description

The agent id to whom this task is assigned.

Returns a string.

## appData

### Description

The application data that is provided to the client with the notification event.

Type: AppData object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," AppData Class Constructor, page 286

## customerName

### Description

The customer username.

Type: string.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## genericconnection

### Description

The generic connection object that associated with this Generic task.

Type: line

## genericId

### Description

The unique generic id generated from an Enqueue. It identifies the data in the database associated with the generic task.

Type: string.

## groupId

### Description

The group id for which task is initiated.

Type: string.

## question

### Description

The question of this generic task which is raised by customer.

Type: string.

## statistics

### Description

Task statistics of the generic task.

Type: object.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## subject

### Description

The subject of the generic task.

Type: string.

## userdata

### Description

User data object that is associated with this generic task.

Type: object.

# GenericTask Class Method

The following is the GenericTask class method.

## gettpUrl

### Syntax

**gettpUrl***(defaultUrl)*

### Description

Returns the URL for the given task for the third-party routing.

### Parameters

| *Parameter* | *Description* |
|---|---|
| defaultUrl | If not null, this value will override the generated base URL. |

### Returns

Returns the URL for the given task.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## getUrl

### Syntax

**getUrl***(defaultUrl)*

### Description

Returns the URL for the given task.

### Parameters

| *Parameter* | *Description* |
|---|---|
| defaultUrl | If not null, this value will override the generated base URL. |

### Returns

Returns the URL for the given task.

# GLOBALS Class Fields

The following are global fields, which may be accessed without instantiating an object. They are accessed as shown.

## A2AChat.PS_JR

### Description

The Journal Routing constant.

### Example

A2AChat.PS_JR = "ps_jr";

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## A2AChat.TYPE_ANSWER

### Description

The constant representing an A2AChat type of answer.

This type is generated when a different user wants this user to answer an A2AChat.

### Example

A2AChat.TYPE_ANSWER = "answer";

## A2AChat.TYPE_CONSULT

### Description

The constant representing an A2AChat type of consult.

This type is generated when this user wants to consult a different user.

### Example

A2AChat.TYPE_CONSULT = "consult";

## Server.TYPE_CTI

### Description

The constant representing a CTI server.

### Example

Server.TYPE_CTI = "CTI";

## Server.TYPE_UQ

### Description

The UQ server type.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Example

Server.TYPE_UQ = "UQ";

## Task.TYPE_A2ACHAT

### Description

The constant representing an A2AChat.

### Example

Task.TYPE_A2ACHAT = "A2ACHAT";

## Task.TYPE_CHAT

### Description

The constant representing a chat task.

### Example

Task.TYPE_CHAT = "CHAT";

## Task.TYPE_CTI

### Description

The constant representing a CTI task.

### Example

Task.TYPE_CTI = "CTI";

## Task.TYPE_EMAIL

### Description

The constant representing an email task.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Example

Task.TYPE_EMAIL = "EMAIL";

## Task.TYPE_GENERIC

### Description

The constant representing a generic task.

### Example

Task.TYPE_GENERIC = "GENERIC";

# GLOBALS Class Methods

The following are the GLOBALS class methods.

## initJSMCAPI

### Syntax

```
initJSMCAPI()
```

### Description

The initialization function of the JSMCAPI. This is the first function should be called by application.

### Parameters

None.

### Returns

None. Initializes JSMCAPI.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# isValid

## Syntax

**isValid***(obj)*

## Description

Returns true if an object is not undefined and not null. *Obj* can be any JavaScript object or literal. This method is a convenience method to check that an object is both not null and not undefined.

## Parameters

| *Parameter* | *Description* |
|---|---|
| *obj* | *Obj* can be any JavaScript object or literal. |

## Returns

Returns a boolean.

Returns true if an object is not undefined and not null.

# MCFBroadcast

## Syntax

**MCFBroadcast***(cluster,queue,task,state,presence,message,securitylevel, importancelevel,senderid,NameValuePairString)*

## Description

Broadcast a message to any queue, cluster, or only agents.

## Parameters

| *Parameter* | *Description* |
|---|---|
| cluster | Cluster Id to which we need to send broadcast message. |
| queue | Denotes the queue Id in the cluster. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Parameter* | *Description* |
|---|---|
| task | Denotes the task, such as email, chat, voice, or generic. |
| state | Denotes the state, such as LoggedIn and NotLoggedIn. |
| presence | Denotes whether the agent is ready or not ready. |
| Message | Enter the message to broadcast. |
| security level | Security level defined by application developers. |
| importance level | Importance level defined by application developers. |
| sender Id | The sender's user id |
| namevaluepairs | Any extra data formed as name-value pair string. |

### Returns

None

# Group Class Constructor

The following is the Group class constructor.

## Group

### Syntax

```
Group()
```

### Description

The Group object describes the group information.

### Parameters

None.

### Returns

A Group object.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Group Class Fields

The following are the Group class fields.

## id

### Description

The group ID.

Type: string.

## name

### Description

The group name.

Type: string.

## registered

### Description

True if the Group is registered on the server.

Type: boolean.

## statistics

### Description

The group statistics for CTI.

Type: GroupStatistics object

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," GroupStatistics1 Class Constructor, page 392

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## statistics1

### Description

The group statistics for the queue server.

Type: GroupStatistics1 object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," GroupStatistics1 Class Constructor, page 392

## statistics2

### Description

The group statistics for the queue server.

Type: GroupStatistics2 object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," GroupStatistics2 Class Constructor, page 396

# Group Class Callback Event Methods

The following are the Group class callback event methods.

## onStat

### Syntax

**onStat***(event)*

### Description

Fires when there is new statistics going to this group.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onStat1

### Syntax

**onStat1***(event)*

### Description

Fires when statistics1 is received.

## onStat2

### Syntax

**onStat2***(event)*

### Description

Fires when statistics2 is received.

## onTaskAdded

### Syntax

**onTaskAdded***(event)*

### Description

Fires when a task added to this group.

## onTaskRemoved

### Syntax

**onTaskRemoved***(event)*

### Description

Fires when a task is removed from this group.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# GroupStatistics Constructor

The following is the GroupStatistics class constructor.

## GroupStatistics

### Syntax

```
GroupStatistics()
```

### Description

The group statistics information.

### Parameters

None.

### Returns

A GroupStatistics object.

# GroupStatistics Fields

The following are the GroupStatistics class fields.

## data

### Description

Key value pairs that include all statistics.

## listOfTasksInTheQueueByTaskType

### Description

Denotes the task type, task state and the time for which the task is in the system.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## maxTaskCompletionTime

### Description

Longest wait time for a task in the queue.

## newestTask

### Description

Time elapsed for the most recent task.

## newestTaskCompletionTime

### Description

Difference between queue time and the time when task is done.

## numberOfAbandoned

### Description

Number of tasks that are abandoned.

## numberOfLoggedIn

### Description

Number of agents that are logged in the queue.

## numberOfQueued

### Description

Number of tasks that are queued.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## numUnassignedTasks

### Description

Number of unassigned tasks.

## queuedWaitTime

### Description

The average wait time, in seconds, of a queued task.

## queueUpTime

### Description

Time since the queue is available on the system (startup/boot time).

## relativeQueueLoad

### Description

Relative queue load.

## timeElapsedOldestTask

### Description

Time elapsed for the oldest task (difference between current time and en-queue time).

# GroupStatistics1 Class Constructor

The following is the GroupStatistics1 class constructor.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## GroupStatistics1

### Syntax

**GroupStatistics1***(data)*

### Description

UQ Group statistics sent on group refresh 1.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *data* | Statistical data from the UQ server. |

### Returns

A GroupStatistics1 object.

# GroupStatistics1 Class Fields

The following are the GroupStatistics1 class fields.

## mostRecentTaskDone

### Description

The most recently done task in the group.

Type: string.

## mostRecentTaskDoneData

### Description

The data for the most recent task done.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## mostRecentTaskEnqueued

### Description

The most recently enqueued task in the group.

Type: string.

## mostRecentTaskEnqueuedData

### Description

The data for the most recently enqueued task.

Type: string.

## numAgentsAvailable

### Description

Number of agents available in the group.

Type: string.

## numAgentsLoggedIn

### Description

Number of agents logged in on the group.

Type: string.

## numEscalation

### Description

Number of escalated tasks in the group.

Type: string.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## numOverflow

### Description

Number of overflowed tasks in the group.

Type: string.

## numTaskAccepted

### Description

Number of tasks accepted in the group.

Type: string.

## numTaskDone

### Description

Number of tasks done in the group.

Type: string.

## numTaskQueued

### Description

Number of tasks queued in the group.

Type: string.

## reasonFlag

### Description

The reason flag for this GroupStatistics event.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## taskTotalTimeInSystem

### Description

The total time in the system.

Type: string.

## timeSinceStart

### Description

The time since start.

Type: string.

# GroupStatistics2 Class Constructor

The following is the GroupStatistics2 class constructor.

## GroupStatistics2

### Syntax

**`GroupStatistics2`***`(data)`*

### Description

UQ Group statistics sent on group refresh 2.

### Parameters

| ***Parameter*** | ***Description*** |
|---|---|
| *data* | UQ server statistical data. |

### Returns

A GroupStatistics2 object.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# GroupStatistics2 Class Fields

The following are the GroupStatistics2 class fields.

## averageTaskDuration

### Description

The average task duration in the group.

Type: string.

## averageWaitTime

### Description

The average wait time in the group.

Type: string.

## oldestTask

### Description

The oldest task in the group.

Type: string.

## recentTask

### Description

The most recent task in the group.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## timeElapsedOldestTask

### Description

The time elapsed for the oldest task in the group.

Type: string.

## timeElapsedRecentTask

### Description

The time elapsed for the most recent task in the group.

# Line Class Constructor

The following is the Line class constructor.

## Line

### Syntax

```
Line()
```

### Description

The Line object describes the line of the extension. JSMCAPI only supports one extension with two lines and two extensions with one line in each.

### Parameters

None.

### Returns

A Line object.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Line Class Fields

The following are the Line class fields.

## call

### Description

The Call object on the line.

Type: Call object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Call Class Constructor, page 292

## caps

### Description

The capabilities of the line.

Type: LineCaps object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," LineCaps Class Constructor, page 427

## connectionid

### Description

The call connection ID on the line.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## id

### Description

The line ID.

Type: string.

## isMuted

### Description

Flag to see whether extension is muted or not.

Type: string.

## state

### Description

The line state.

Type: string with the following constants:

| *Value* | *Description* |
|---|---|
| ST_DIALING | The dialing state. |
| ST_DROPPED | The dropped state. |
| ST_HELD | The held state. |
| ST_IDLE | The idle state. |
| ST_OFFHOOK | The offhook state. |
| ST_RINGING | The ringing state. |
| ST_TALKING | The talking state. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Line Class Methods

The following are the Line class methods.

## alternate

### Syntax

**alternate***(reason)*

### Description

Alternate a call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to alternate a call. |

### Returns

Request number.

## answer

### Syntax

**answer***(reason)*

### Description

Answer a call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to answer a call. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Request number.

## attachUserData

### Syntax

**attachUserDate**(*userdata,reason)*

### Description

Attach user data to a call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *userdata* | The user data to attach to the call. |
| *reason* | The reason to attach the user data. |

### Returns

Request number.

## clear

### Syntax

**clear**(*reason)*

### Description

Clear a conference call.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to clear the conference. |

### Returns

Request number.

## complete

### Syntax

**complete***(reason)*

### Description

Complete the two-step transfer/conference.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to complete the two-step transfer/conference. |

### Returns

Request number.

## conference

### Syntax

**conference***(destination,reason,userdata,calldata)*

### Description

Conference a call.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *destination* | The destination being invited into the conference. |
| *reason* | The reason to conference. |
| *userdata* | The user data to be attached. |
| *calldata* | The call data to be modified. |

### Returns

Request number.

## conferenceSingle

### Syntax

**`conferenceSingle`***`(destination,reason,userdata,calldata)`*

### Description

Single-step conference a call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *destination* | The destination being invited into the conference. |
| *reason* | The reason to conference. |
| *userdata* | The user data to be attached. |
| *calldata* | The call data to be modified. |

### Returns

Request number.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## dial

### Syntax

**dial**(*number,reason,userdata*)

### Description

Dial out.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *destination* | The destination being invited into the conference. |
| *reason* | The reason to conference. |
| *userdata* | The user data to be attached. |

### Returns

Request number.

## dropParty

### Syntax

**dropParty**(*destination,reason*)

### Description

Drop a party in conference.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *destination* | The destination being dropped from the conference. |
| *reason* | The reason to drop the party. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Request number.

## getAni

### Syntax

```
getAni()
```

### Description

Get the ANI.

### Parameters

None.

### Returns

Returns a string representing the ANI.

## getDescr

### Syntax

```
getDescr()
```

### Description

Get the description attached to the call on the line.

### Parameters

None.

### Returns

Returns a string.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## getDnis

### Syntax

**getDnis**()

### Description

Get the DNIS.

### Parameters

None.

### Returns

Returns a string.

## getPadvalue

### Syntax

**getPadvalue***(key)*

### Description

Get the PAD value.

### Returns

A string representing PAD value.

## getReferenceId

### Syntax

**getReferenceId**()

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Get the reference ID attached to the call on the line.

### Parameters

None.

### Returns

Returns a string.

## getUrl

### Syntax

**getUrl***(defaultUrl)*

### Description

Get the URL for screen pop-up.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *defaultUrl* | The default pop-up URL. |

### Returns

Returns a string.

## grabCall

### Syntax

**grabCall***(destination,reason)*

### Description

Grab a call from the queue.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *destination* | The destination to be grabbed. |
| *reason* | The reason for the grab. |

### Returns

Request number.

## hold

### Syntax

**hold***(reason)*

### Description

Hold a call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to hold the call. |

### Returns

Request number.

## join

### Syntax

**join***(reason,conferenceId)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Join an existing call/conference.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to join the call. |
| conferenceId | Call id. |

### Returns

Request number.

## mute

### Syntax

**mute***(reason)*

### Description

Mute an extension.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to mute the call. |

### Returns

Request number.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## park

### Syntax

**park**(*destination,reason)*

### Description

Park a call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *destination* | The destination on which the call will be parked. |
| *reason* | The reason to park the call. |

### Returns

Request number.

## reconnect

### Syntax

**reconnect**(*reason)*

### Description

Reconnect to the original party during two-step transfer/conference.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to reconnect. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Request number.

## reject

### Syntax

**reject**(*reason*)

### Description

Reject an incoming call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to reject. |

### Returns

Request number.

## release

### Syntax

**release**(*reason*)

### Description

Release a call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to release the call. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Request number.

## retrieve

### Syntax

**retrieve***(reason)*

### Description

Retrieve a held call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to retrieve the call. |

### Returns

Request number.

## sendDTMF

### Syntax

Syntax

**sendDTMF***(reason,stringDTMF)*

### Description

Send DTMF tones to the switch.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason for requesting DTMF. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Parameter | Description |
|---|---|
| *stringDTMF* | DTMF tones string (0-9,*,#). |

### Returns

Request number.

## setcallresult

### Syntax

**setcallresult***(result)*

### Description

Set call result.

### Parameters

| Parameter | Description |
|---|---|
| *result* | Result of the call. |

### Returns

Request number.

## setcallresultDNC

### Syntax

**setcallresultDNC***(number,reason)*

### Description

Do not call.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *number* | The number not to be disturbed. |
| *reason* | The reason not to disturb. |

### Returns

Request number.

## setcallresultReschedule

### Syntax

**setcallresultReschedule***(hour,minutes,day,month,year)*

### Description

Reschedule a call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *hour* | Hours |
| *Minutes* | Minutes |
| *Day* | Day |
| *Month* | Month |
| *Year* | Year |

### Returns

Request number.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# transfer

## Syntax

**transfer***(destination,reason,userdata,calldata)*

## Description

Transfer a call.

## Parameters

| *Parameter* | *Description* |
|---|---|
| *destination* | The destination to which the call will be transferred. |
| *reason* | The reason to transfer. |
| *userdata* | The user data to be attached. |
| *calldata* | The call data to be modified. |

## Returns

Request number.

# transferMute

## Syntax

**transferMute***(destination,reason,userdata,calldata)*

## Description

Transfer mute a call.

## Parameters

| *Parameter* | *Description* |
|---|---|
| *destination* | The destination to which the call will be transferred. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Parameter* | *Description* |
|---|---|
| *reason* | The reason to transfer. |
| *userdata* | The user data to be attached. |
| *calldata* | The call data to be modified. |

### Returns

Request number.

## unmute

### Syntax

**unmute***(reason)*

### Description

Unmute a call.

### Parameters

| *Parameter* | *Description* |
|---|---|
| reason | The reason to unmute a call. |

### Returns

Request number.

## updateCallData

### Syntax

**updateCallData***(calldata,reason)*

### Description

Update call data to a call.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *calldata* | The call data to update. |
| *reason* | The reason to update. |

### Returns

Request number.

## Line Class Callback Event Methods

The following are the Line class callback event methods.

## onAlternating

### Syntax

**onAlternating***(event)*

### Description

Fires when alternating the call.

## onAnswering

### Syntax

**onAnswering***(event)*

### Description

Fires when answering the call.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onAttachingUD

### Syntax

**onAttachingUD***(event)*

### Description

Fires when attaching user data.

## onCallDataChanged

### Syntax

**onCallDataChanged***(event)*

### Description

Fires when the call data on the call is changed.

## onCapabilitiesChanged

### Syntax

**onCapabilitiesChanged***(event)*

### Description

Fires when the capabilities changed.

## onClearing

### Syntax

**onClearing***(event)*

### Description

Fires when clearing the conference.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onCompleting

### Syntax

**onCompleting***(event)*

### Description

Fires when completing the two-step transfer/conference.

## onConferencing

### Syntax

**onConferencing***(event)*

### Description

Fires when conferencing the call.

## onDialing

### Syntax

**onDialing***(event)*

### Description

Fires when there is an outgoing call.

## onDropped

### Syntax

**onDropped***(event)*

### Description

Fires when the call is released.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onError

### Syntax

**onError***(event)*

### Description

Fires when there is error.

## onGrabbing

### Syntax

**onGrabbing***(event)*

### Description

Fires when grabbing a call from a queue.

## onHeld

### Syntax

**onHeld***(event)*

### Description

Fires when the call is on hold.

## onHolding

### Syntax

**onHolding***(event)*

### Description

Fires when holding the call.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onJoining

### Syntax

**onJoining***(event)*

### Description

Fires when joining a call or a conference.

## onMuted

### Syntax

**onMuted***(event)*

### Description

Fires when a call is muted.

## onOffHook

### Syntax

**onOffHook***(event)*

### Description

Fires when the line is off hook.

## onOnHook

### Syntax

**onOnHook***(event)*

### Description

Fires when the line is on hook.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onParking

### Syntax

**onParking***(event)*

### Description

Fires when parking the call.

## onPartyAdded

### Syntax

**onPartyAdded***(event)*

### Description

Fires when a new call party coming.

## onPartyChanged

### Syntax

**onPartyChanged***(event)*

### Description

Fires when the call party has changed.

## onPartyRemoved

### Syntax

**onPartyRemoved***(event)*

### Description

Fires when a call party is removed.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onReconnecting

### Syntax

**onReconnecting***(event)*

### Description

Fires when reconnecting the call.

## onRejected

### Syntax

**onRejected***(event)*

### Description

Fires when call is rejected.

## onRejecting

### Syntax

**onRejecting***(event)*

### Description

Fires agent rejects an incoming call.

## onReleasing

### Syntax

**onReleasing***(event)*

### Description

Fires when releasing the call.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onRetrieving

### Syntax

**onRetrieving***(event)*

### Description

Fires when retrieving the call.

## onRinging

### Syntax

**onRinging***(event)*

### Description

Fires when there is an incoming call.

## onSetcallresult

### Syntax

**onSetcallresult***(event)*

### Description

Fires when there is a request for setting call result.

## onSetcallresultDNC

### Syntax

**onSetcallresultDNC***(event)*

### Description

Fires when there is a request for do not call.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onSetcallresultReschedule

### Syntax

**onSetcallresultReschedule***(event)*

### Description

Fires when there is a request to reschedule a call.

## onTalking

### Syntax

**onTalking***(event)*

### Description

Fires when the call is established.

## onTransfering

### Syntax

**onTransfering***(event)*

### Description

Fires when transferring the call.

## onUnmuted

### Syntax

**onUnmuted***(event)*

### Description

Fires when call is unmuted.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onUpdatingCD

### Syntax

**onUpdatingCD***(event)*

### Description

Fires when updating the call data.

## onUserDataChanged

### Syntax

**onUserDataChanged***(event)*

### Description

Fires when the user data attached on the call is changed.

# LineCaps Class Constructor

The following is the LineCaps class constructor.

## LineCaps

### Syntax

**LineCaps***(strCaps)*

### Description

Describes the line's capabilities.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *strCaps* | The line's capabilities. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

A LineCaps object.

## LineCaps Class Fields

The following are the LineCaps class fields.

### canAlternate

#### Description

The line has alternate capability.

Type: boolean.

### canAnswer

#### Description

The line has answer capability.

Type: boolean.

### canAttachUserDate

#### Description

The line has attach user data capability.

Type: boolean.

### canClear

#### Description

The line has clear capability.

Type: boolean.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## canComplete

### Description

The line has complete capability.

Type: boolean.

## canConference

### Description

The line has conference capability.

Type: boolean.

## canConferenceSingle

### Description

The line has conference single capability.

Type: boolean.

## canDropParty

### Description

The line has drop party capability.

Type: boolean.

## canHold

### Description

The line has hold capability.

Type: boolean.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## canMute

### Description

The line has mute capability.

Type: boolean.

## canPark

### Description

The line has park capability.

Type: boolean.

## canReconnect

### Description

The line has reconnect capability.

Type: boolean.

## canReject

### Description

The line has reject capability.

Type: boolean.

## canRelease

### Description

The line has release capability.

Type: boolean.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## canRetrieve

### Description

The line has retrieve capability.

Type: boolean.

## canSendDTMF

### Description

The line has DTMF capability.

Type: boolean.

## canSetcallresult

### Description

The line has set call result capability.

Type: boolean.

## canSetcallresultDNC

### Description

The line has do not call capability.

Type: boolean.

## canSetcallresultReschedule

### Description

The line has reschedule capability.

Type: boolean.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## canTransfer

### Description

The line has transfer capability.

Type: boolean.

## canTransferMute

### Description

The line has transfer mute capability.

Type: boolean.

## canUnmute

### Description

The line has unmute capability.

Type: boolean.

## canUpdateCallData

### Description

The line has update call data capability.

Type: boolean.

# MCEvent Class Constructor

The following is the MCEvent class constructor.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## MCEvent

### Syntax

```
MCEvent()
```

### Description

The MCEvent will be passed to the application event handler.

### Parameters

None.

### Returns

An MCEvent object.

# MCEvent Class Fields

The following are the MCEvent class fields.

## extension

### Description

The extension associated with the event.

Type: Extension object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Extension Class Constructor, page 348

## group

### Description

The group associated with the event.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Type: Group object.

***See Also***

Chapter 13, "Understanding JSMCAPI Classes," Group Class Constructor, page 386

## reason

### Description

The reason associated with the event.

Type: Reason object.

***See Also***

Chapter 13, "Understanding JSMCAPI Classes," Reason Class Constructor, page 442

## user

### Description

The user associated with the event.

Type: User object.

***See Also***

Chapter 13, "Understanding JSMCAPI Classes," User Class Constructor, page 469

# MediaType Class Constructor

The following is the MediaType class constructor.

## MediaType

### Syntax

```
MediaType()
```

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Distinguishes between email and CTI media types.

### Parameters

None.

### Returns

A MediaType object.

## MediaType Class Field

The following is the MediaType field.

### type

#### Description

The media type.

Type: string with the following constants:

| *Value* | *Description* |
|---|---|
| MT_CHAT | The chat media type. |
| MT_EMAIL | The email media type. |
| MT_GENERIC | The generic media type. |
| MT_VOICE | The voice media type (CTI). |

## PSMC Class Constructor

The following is the PSMC class constructor.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## PSMC

### Syntax

**PSMC**()

### Description

The global object that the application can access.

### Parameters

None.

### Returns

A PSMC object.

# PSMC Class Fields

The following are the PSMC class fields.

## renserver

### Description

The PSMC instance of the RenServer object.

Type: RenServer object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," RenServer Class Constructor, page 444

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## servers

### Description

An associative array the different server objects, indexed by server ID.

Type: associative array.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Server Class Constructor, page 446

## sessions

### Description

An associative array that holds the different session objects, indexed by server ID.

Type: associative array.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Session Class Constructor, page 450

# PSMC Class Methods

The following are the PSMC class methods.

## closeSession

### Syntax

**closeSession***(serverId)*

### Description

Closes the specified session and server object and deletes them.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *serverId* | The ID of the REN server cluster for a UQ server, or the server ID for a CTI server. |

### Returns

None.

## getCallById

### Syntax

**`getCallById`***`(id)`*

### Description

Retrieve the call object by the call ID.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *id* | The call ID. |

### Returns

Type: Call object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Call Class Constructor, page 292

## getChatById

### Syntax

**`getChatById`***`(id)`*

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Retrieve the chat object by the task id.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *id* | The task ID. |

### Returns

Type: chat object.

## getEmailById

### Syntax

**getEmailById***(id)*

### Description

Retrieve the email object by the task id.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *id* | The task ID. |

### Returns

Type: email object.

## getGenericTaskById

### Syntax

**getGenericTaskById***(id)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Retrieve the Generic object by the task id.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *id* | The task ID. |

### Returns

Type: generic object.

## getLineById

### Syntax

**getLineById***(id)*

### Description

Retrieve the line object by the line ID.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *id* | The line ID. |

### Returns

Type: Line object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," Line Class Constructor, page 398

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## openSession

### Syntax

**openSession***(id)*

### Description

Creates the session and server objects for the PSMC.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *id* | The ID of the REN server cluster for a UQ server, or the server ID for a CTI server. |

### Returns

None.

## start

### Syntax

**start**()

### Description

Start the JSMCAPI.

### Parameters

None.

### Returns

None.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### stop

#### Syntax

**stop**()

#### Description

Stop the JSMCAPI.

#### Parameters

None.

#### Returns

None.

## Reason Class Constructor

The following is the Reason class constructor.

### Reason

#### Syntax

**reason**(*code,desc,reasondata1,reasondata2,reasondata3*)

#### Description

The Reason object carries the reason code and reason description.

#### Parameters

| *Parameter* | *Description* |
|---|---|
| *code* | The reason code. |
| *desc* | The reason description. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Parameter* | *Description* |
|---|---|
| reasondata1 | The reason data1. |
| reasondata2 | The reason data2. |
| reasondata3 | The reason data3 |

### Returns

Type: Reason object.

## Reason Class Fields

The following are the Reason class fields.

### code

#### Description

The reason code.

Type: string.

### desc

#### Description

The reason description.

Type: string.

### reasonData1

#### Description

Place holder for extra data.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## reasonData2

### Description

Place holder for extra data.

Type: string.

## reasonData3

### Description

Place holder for extra data.

Type: string.

# RenServer Class Constructor

The following is the RenServer class constructor.

## RenServer

### Syntax

```
RenServer()
```

### Description

The RenServer object describes the REN server cluster information.

### Parameters

None.

### Returns

A RenServer object.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# RenServer Class Fields

The following are the RenServer class fields.

## isRunning

### Description

Flag containing the state of the REN server connection.

Type: boolean.

## url

### Description

The REN server URL.

Type: string.

# RenServer Class Callback Event Methods

The following are the RenServer class callback event methods.

## onDown

### Syntax

**onDown***(event)*

### Description

Fires when the REN server is down.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### onUp

#### Syntax

**onUp***(event)*

#### Description

Fires when the REN server is up.

## Server Class Constructor

The following is the Server class constructor.

### Server

#### Syntax

**Server***(serverId)*

#### Description

The Server object describes the server information.

#### Parameters

| *Parameter* | *Description* |
|---|---|
| *serverId* | The ID of the REN server cluster for a UQ server, or the server ID for a CTI server. |

#### Returns

A Server object.

## Server Class Fields

The following are the Server class fields.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## id

### Description

The server ID.

Type: string.

## info

### Description

The server information string.

Type: string.

## state

### Description

The server state.

Type: string with the following constants.

| *Value* | *Description* |
|---|---|
| ST_INSERVICE | The server is in service. |
| ST_OUTOFSERVICE | The server is out of service. |

## type

### Description

The type of server, either CTI or UQ.

Type: string with the following constants.

| *Value* | *Description* |
|---|---|
| TYPE_CTI | The server is a CTI server. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Value* | *Description* |
|---|---|
| TYPE_MCS | The server is a third-party MultiChannel server. |
| TYPE_UQ | The server is a queue server. |

### Example

The TYPE_* fields are global fields, which may be accessed without instantiating an object. They are accessed as shown in this example:

```
Server.TYPE_UQ = "UQ";
```

### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," GLOBALS Class Fields, page 381

# Server Class Callback Event Methods

The following are the Server class callback event methods.

## onBroadcast

### Syntax

**onBroadcast***(event)*

### Description

Fires when there is broadcast message.

## onHbLost

### Syntax

**onHbLost***(event)*

### Description

Fires when the server's heartbeat is lost.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onHbRecovered

### Syntax

**onHbRecovered***(event)*

### Description

Fires when the server's heartbeat is recovered.

## onInService

### Syntax

**onInService***(event)*

### Description

Fires when the server changes state to in service.

## onOutOfService

### Syntax

**onOutOfService***(event)*

### Description

Fires when the server changes state to out of service.

## onRestart

### Syntax

**onRestart***(event)*

### Description

Fires when server restarts.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Session Class Constructor

The following is the Session class constructor.

## Session

### Syntax

**Session***(serverId)*

### Description

The Session object describes the session with the server.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *serverId* | The ID of the REN server cluster for a UQ server, or the server ID for a CTI server. |

### Returns

Returns a Session object.

# Session Class Fields

The following are the Session class fields.

## addresses

### Description

The associative array of addresses that the session register.

Type: associative array.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## buddies

### Description

The buddy hash table in this session.

Type: associative array.

## groups

### Description

The group hash table in this session.

Type: associative array.

## id

### Description

The session ID generated by the server.

Type: string.

## intervalBetweenReqs

### Description

Interval in milliseconds between requests.

Type: number

## numberRegsPerBulkReq

### Description

Number of users or groups to register for one bulk register request.

Type: number

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## serverId

### Description

The serverId will be unique for every session. It is used to lookup the protocol objects.

Type: string.

## state

### Description

The session state.

Type: string with the following constants.

| *Value* | *Description* |
|---|---|
| ST_ACTIVE | The session is in the active state. |
| ST_CLOSED | The session is in the closed state. |
| ST_CLOSING | The session is in the closing state. |
| ST_IDLE | The session is in the idle state. |

## user

### Description

The current user for the session.

Type: User object.

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," User Class Constructor, page 469

# Session Class Methods

The following are the Session class methods.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## broadcastSubscribe

### Syntax

**broadcastSubscribe**(*cluster,queue,task,state,presence,method)*

### Description

Subscribe to broadcast messages.

### Returns

Object.

## broadcastUnsubscribe

### Syntax

**broadcastunsubscribe**(*type)*

### Description

Unsubscribe to broadcast messages.

### Returns

None.

## close

### Syntax

**close**()

### Description

Close the session with the server.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

None.

### Returns

Request number.

## open

### Syntax

**open**( )

### Description

Open the session with the server.

### Parameters

None.

### Returns

Request number.

## registerAddress

### Syntax

**registerAddress***(address)*

### Description

Register the address to the server so that the server will report all events that occur on this address.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *address* | The address to be registered. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Request number.

## registerBuddy

### Syntax

**registerBuddy***(buddy)*

### Description

Register the buddy to the JSMCAPI so that the JSMCAPI will report all state events that occur on this buddy.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *buddy* | The buddy to be registered. |

### Returns

Request number.

## registerBuddiesBulk

### Syntax

**registerBuddiesBulk***(buddies,numRegsPerBulkReq,intervalBetweenReqs)*

### Description

Register multiple buddies to the JSMCAPI so that the JSMCAPI will report state events that occur for the buddies.

### Parameters

| *Parameter* | *Description* |
|---|---|
| buddies | An array of buddies |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Parameter | Description |
|---|---|
| numRegPerBulkReq | Number of buddies to register in one bulk registration request. |
| intervalBetweenReqs | Interval in milliseconds between requests. |

### Returns

Request number.

## registerGroup

### Syntax

**registerGroup***(group)*

### Description

Register the group to the server so that the server will report all events that occur on this group.

### Parameters

| Parameter | Description |
|---|---|
| *group* | The group to be registered. |

### Returns

Request number.

## registerGroupsBulk

### Syntax

**registerGroupsBulk***(groups,numRegsPerBulkReq,intervalBetweenReqs)*

### Description

Register multiple groups to the server so that the server will report all events that occur on the groups.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| groups | An array of groups. |
| numRegPerBulkReq | Number of groups to register in one bulk registration request. |
| intervalBetweenReqs | Interval in milliseconds between requests. |

### Returns

Request number.

## registerUser

### Syntax

**registerUser***(user)*

### Description

Register the user to the server so that the server will report all those events that happen on this user.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *user* | The user to be registered. |

### Returns

Request number.

## setAutoRecovery

### Syntax

**setAutoRecovery***(autorecover)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Sets the auto recovery feature of the queue server connection to off.

### Parameters

| *Parameter* | *Description* |
|---|---|
| autoRecover | Recover. |

## statPublish

### Syntax

**statPublish***(userStat,groupStat)*

### Description

Sets the statistics publishing levels for the queue server.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *userStat* | Either 0, 1, or 2 for UserStatistics publishing.<br>Enter 0 for no statistics publishing, 1 for Statistics1 publishing, and 2 for Statistics2 and Statistic1 publishing. |
| *groupStat* | Either 0, 1, or 2 for GroupStatistics publishing.<br>Enter 0 for no statistics publishing, 1 for Statistics1 publishing, and 2 for Statistics2 and Statistic1 publishing. |

### Returns

Request number.

## unregisterAddress

### Syntax

**unRegisterAddress***(address)*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Unregister the address from the server so that the server will not report any events about this address.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *address* | The address to be unregistered. |

### Returns

Request number.

## unregisterBuddy

### Syntax

**unregisterBuddy***(buddy)*

### Description

Unregister the buddy from the JSMCAPI so that the JSMCAPI will not report any events about the buddy state change.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *buddy* | The buddy to be unregistered. |

### Returns

Request number.

## unregisterGroup

### Syntax

**unregisterGroup***(group)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Unregister the group from the server so that the server will not report any events about this group.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *group* | The group to be unregistered. |

### Returns

Request number.

## unregisterUser

### Syntax

**unregisterUser***(user)*

### Description

Unregister the user from the server so that the server will not report any events to this user.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *user* | The user to be unregistered. |

### Returns

Request number.

# Session Class Callback Event Methods

The following are the Session class callback event methods.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onAddressRegistered

### Syntax

**onAddressRegistered***(event)*

### Description

Fires when the address is registered by the session.

## onAddressUnregistered

### Syntax

**onAddressUnregistered***(event)*

### Description

Fires when the address is unregistered by the session.

## onBuddyRegistered

### Syntax

**onBuddyRegistered***(event)*

### Description

Fires when the buddy is registered by the session.

## onBuddyUnregistered

### Syntax

**onBuddyUnregistered***(event)*

### Description

Fires when the buddy is unregistered by the session.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onClosed

### Syntax

**onClosed***(event)*

### Description

Fires when the session is closed.

## onError

### Syntax

**onError***(event)*

### Description

Fires when there is a session error.

## onGroupRegistered

### Syntax

**onGroupRegistered***(event)*

### Description

Fires when a group is registered by the session.

## onGroupUnregistered

### Syntax

**onGroupUnregistered***(event)*

### Description

Fires when a group is unregistered by the session.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onInfo

### Syntax

**onInfo***(event)*

### Description

Fires when there is a session information event, such as the user is already logged in.

## onOpened

### Syntax

**onOpened***(event)*

### Description

Fires when the session is opened.

## onUserRegistered

### Syntax

**onUserRegistered***(event)*

### Description

Fires when the user is registered by the session.

## onUserUnregistered

### Syntax

**onUserUnregistered***(event)*

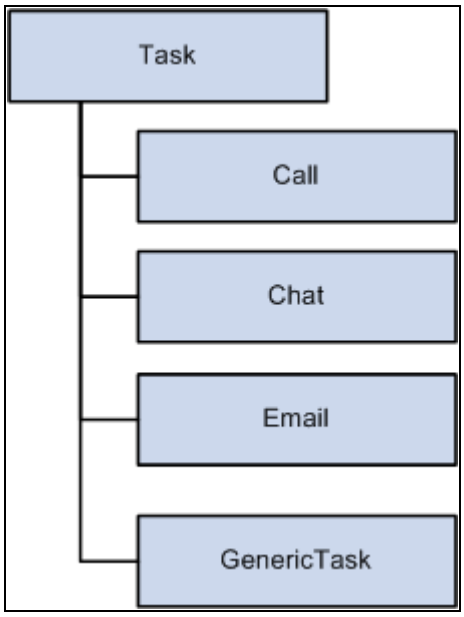
### Description

Fires when the user gets unregistered by the session.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Task Class Hierarchy

The Task class can be extended by other subclasses. The following flow chart shows the different subclasses and how they interrelate.




Task class hierarchy

***See Also***

Chapter 13, "Understanding JSMCAPI Classes," Call Class Constructor, page 292

Chapter 13, "Understanding JSMCAPI Classes," Chat Class Constructor, page 298

Chapter 13, "Understanding JSMCAPI Classes," Email Class Constructor, page 327

Chapter 13, "Understanding JSMCAPI Classes," GenericAddress Class Constructor, page 360

# Task Class Constructor

The following is the Task class constructor.

## Task

### Syntax

```
Task()
```


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Task is an abstract base class.

### Parameters

None.

### Returns

Type: Task object.

## Task Class Fields

The following are the Task class fields.

### caseid

#### Description

The associated case ID.

Type: string.

### cost

#### Description

The associated cost.

Type: string.

### customerid

#### Description

The associated customer ID.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## group

### Description

The group for which task is assigned.

Type: string.

## id

### Description

The task ID.

Type: string.

## onStat

### Description

Method that will be triggered when there is statistics published regarding this task.

Type: string.

## priority

### Description

The priority of the task.

Type: string.

## type

### Description

The type of the task.

Type: string, one of the following:


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Value* | *Description* |
|---|---|
| TYPE_A2ACHAT | The A2AChat type. |
| TYPE_CHAT | The chat type. |
| TYPE_CTI | The CTI type. |
| TYPE_EMAIL | The email type. |
| TYPE_GENERIC | The generic type. |

### Example

The TYPE_* fields are global fields, which may be accessed without instantiating an object. They are accessed as shown in this example:

```
Task.TYPE_GENERIC = "GENERIC";
```

#### *See Also*

Chapter 13, "Understanding JSMCAPI Classes," GLOBALS Class Fields, page 381

## urlAbs

### Description

The absolute URL that is useful in constructing URL for new window.

Type: urlAbs object.

## urlRel

### Description

The relative URL that is useful in constructing URL for new window.

Type: urlRel object.

# TaskStatistics Class Constructor

The following is the TaskStatistics class constructor.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

### TaskStatistics

#### Syntax

```
TaskStatistics()
```

#### Description

Describes the task statistics information.

#### Returns

Task statistics object.

## TaskStatistics Class Fields

The following are the TaskStatistics class fields.

### data

#### Description

Key value pairs that includes all statistics.

Type: collection.

### holdTime

#### Description

Time duration that the task is on hold.

Type: number

### queueTime

#### Description

Time duration in the queue for this task.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Type: number

### talkTime

#### Description

Time duration that the task is established.

Type: number

## User Class Constructor

The User class extends the _User class.

See Chapter 13, "Understanding JSMCAPI Classes," _User Class Constructor, page 271.

The following is the User class constructor.

### User

#### Syntax

**User***(id,name,agentId,agentPassword)*

#### Description

Describes the user/agent.

#### Parameters

| ***Parameter*** | ***Description*** |
|---|---|
| *id* | The user ID. |
| *name* | The user name. |
| *agentId* | The agent ID. |
| *agentPassword* | The agent's password. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

A User object.

## User Class Fields

The User class inherits the following fields from the _User class:

- agentId
- caps
- id
- name
- presences
- ST_LOGGEDIN
- ST_LOGGEDOUT
- ST_NOTREADY
- ST_READY
- ST_UNKNOWN
- ST_WORKNOTREADY
- ST_WORKREADY
- states
- statistics
- statistics1
- statistics2

The following are the User class fields.

## addresses

### Description

The addresses associated with this agent.

Type: associative array.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## agentPassword

### Description

The password for this agent.

Type: string.

## language

### Description

The language for this agent.

Type: string.

## registeredAddresses

### Description

The registered addresses for this user.

Type: associative array.

## statesUq

### Description

States of the user for the queue server indexed by group number

Type: object.

# User Class Methods

The following are the User class methods.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

# ctiBusyUq

## Syntax

**ctiBusyUq**(*busy,subtractcost,task)*

## Description

Flip ctiBusy flag, reduce cost of a CTI task from an agent's workload, or both. If the task is the same as the agent's assigned task, then the task's cost will be used. In case they do not match, the task type is checked and the default cost of the task is used to recalculate the workload.

## Parameters

| *Parameter* | *Description* |
|---|---|
| *busy* | The ctibusy flag that can have a value of 0 or 1. |
| *subtractcost* | Flag to reduce cost of the cti task from the agent's workload. It can be set to 0 or 1. |
| *task* | The task object. |

## Returns

Request number.

# disableMedia

## Syntax

**disableMedia**(*group,mediaType,extension,reason)*

## Description

Disable a media channel for an agent on a group/queue.

## Parameters

| *Parameter* | *Description* |
|---|---|
| *group* | The group (queue) on which to disable the specified media type. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Parameter* | *Description* |
|---|---|
| *mediaType* | The media type to disable. |
| *extension* | The extension on which to disable the specified media type. |
| *reason* | The reason for disabling this media type. |

### Returns

Request number.

## enableMedia

### Syntax

**enableMedia**(*group,mediaType,extension,reason*)

### Description

Enable a media channel for an agent on a group/queue.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *group* | The group (queue) on which to enable the specified media type. |
| *mediaType* | The media type to enable. |
| *extension* | The extension on which to enable the specified media type. |
| *reason* | The reason for enabling this media type. |

### Returns

Request number.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# isMediaEnabled

## Syntax

**isMediaEnabled***(mediaType,group)*

## Description

Determines if the specified media type is enabled.

## Parameters

| *Parameter* | *Description* |
|---|---|
| *mediaType* | Enter the media type. |
| *group* | Enter the group. |

## Returns

Returns true if the media type is enabled.

# login

## Syntax

**login***(group,extension,reason)*

## Description

Log the user in to a queue.

## Parameters

| *Parameter* | *Description* |
|---|---|
| *group* | The group to which the user will log in. |
| *extension* | The extension. |
| *reason* | The reason for the login. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Request number.

## loginUq

### Syntax

**loginUq***(group)*

### Description

Log a user in to a UQ server queue.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *group* | The group (queue) to which the user will login. |

### Returns

Request number.

## logout

### Syntax

**logout***(group,extension,reason)*

### Description

Log the user out of the specified group (queue).

### Parameters

| *Parameter* | *Description* |
|---|---|
| *group* | The group from which the user will log out. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Parameter | Description |
|---|---|
| *extension* | The extension. |
| *reason* | The reason for the log out. |

### Returns

Request number.

## logoutUq

### Syntax

**logoutUq***(group)*

### Description

Log the user out from the UQ server queue.

### Parameters

| Parameter | Description |
|---|---|
| *group* | The group (queue) from which to log the user out. |

### Returns

Request number.

## register

### Syntax

**register***(extension,reason)*

### Description

Register an extension for the user.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *extension* | The extension to be registered. |
| *reason* | The reason to register the extension. |

### Returns

Request number.

## setNotReady

### Syntax

**`setNotReady`**`(group,presence,extension,reason)`

### Description

Set the user state as not ready.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *group* | The group on which the user is changing state. |
| *presence* | The presence in the new state. |
| *extension* | The extension with which the user is changing state. |
| *reason* | The reason to set not ready. |

### Returns

Request number.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# setPresence

## Syntax

**setPresence***(group,presence,extension,reason)*

## Description

Set the presence for the current state.

## Parameters

| *Parameter* | *Description* |
|---|---|
| *group* | The group on which the user is changing presence. |
| *presence* | The presence for the state. |
| *extension* | The extension with which the user is changing presence. |
| *reason* | The reason to set the presence. |

## Returns

Request number.

# setPresenceUq

## Syntax

**setPresenceUq***(group,presenceText,reason)*

## Description

Set the Presence for UQ servers.

## Parameters

| *Parameter* | *Description* |
|---|---|
| *group* | The group (queue) on which the user is changing his or her presence. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Parameter | Description |
|---|---|
| *presenceText* | The new presence. |
| *reason* | The reason to set the new presence. |

### Returns

Request number.

## setReady

### Syntax

**setReady**(*group,presence,extension,reason*)

### Description

Set the user state as ready.

### Parameters

| Parameter | Description |
|---|---|
| *group* | The group on which the user is changing state. |
| *presence* | The presence for the state. |
| *extension* | The extension with which the user is changing state. |
| *reason* | The reason to set the state. |

### Returns

Request number.

## setWorkNotReady

### Syntax

**setWorkNotReady**(*group,presence,extension,reason*)

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Set the user state as work not ready.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *group* | The group on which the user is changing state. |
| *presence* | The presence for the state. |
| *extension* | The extension with which the user is changing state. |
| *reason* | The reason to set work not ready. |

### Returns

Request number.

## setWorkReady

### Syntax

**`setWorkReady`**`(group,presence,extension,reason)`

### Description

Set the user state as work ready.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *group* | The group on which the user is changing state. |
| *presence* | The presence for the state. |
| *extension* | The extension with which the user is changing state. |
| *reason* | The reason to set work ready. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Returns

Request number.

## unregister

### Syntax

**unregister***(extension,reason)*

### Description

Unregister an extension for the user.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *extension* | The extension to be unregistered. |
| *reason* | The reason to unregister the extension. |

### Returns

Request number.

# User Class Callback Event Methods

The following are the User class callback event methods.

## onCapabilitiesChanged

### Syntax

**onCapabilitiesChanged***(event)*

### Description

Fires when capabilities changed.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onCtiBusy

### Syntax

**onCtiBusy***(event)*

### Description

Fires when the UQ server is notified that the CTI is busy.

## onCTIBUSYUq

### Syntax

**onCTIBUSYUq***(event)*

### Description

Fires when the UQ server is notified that the CTI is busy.

## onCtiClear

### Syntax

**onCtiClear***(event)*

### Description

Fires when the UQ server is notified that the CTI is clear.

## onDropped

### Syntax

**onDropped***(event)*

### Description

Fires when a user is dropped.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onError

### Syntax

**onError***(event)*

### Description

Fires when an error occurs.

## onInfo

### Syntax

**onInfo***(event)*

### Description

Fires when an information event occurs. Currently only used on UQ server.

## onLoggedIn

### Syntax

**onLoggedIn***(event)*

### Description

Fires when the user is logged in.

## onLoggedOut

### Syntax

**onLoggedOut***(event)*

### Description

Fires when the user is logged out.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onLoggingIn

### Syntax

**onLoggingIn***(event)*

### Description

Fires when the user is logging in.

## onLoggingOut

### Syntax

**onLoggingOut***(event)*

### Description

Fires when the user is logging out.

## onMediaDisabled

### Syntax

**onMediaDisabled***(event)*

### Description

Fires when a media type is disabled.

## onMediaEnabled

### Syntax

**onMediaEnabled***(event)*

### Description

Fires when a media type is enabled.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onNotReady

### Syntax

**onNotReady***(event)*

### Description

Fires when a user state changes to not ready.

## onPresenceChanged

### Syntax

**onPresenceChanged***(event)*

### Description

Fires when the user's presence is changed.

## onReady

### Syntax

**onReady***(event)*

### Description

Fires when the user's state changes to ready.

## onRegistered

### Syntax

**onRegistered***(event)*

### Description

Fires when the address is registered.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## onRegistering

### Syntax

**onRegistering***(event)*

### Description

Fires when registering the address.

## onSettingNotReady

### Syntax

**onSettingNotReady***(event)*

### Description

Fires when setting the user's state to not ready.

## onSettingPresence

### Syntax

*(event)*

**onSettingPresence**

### Description

Fires when setting presence for the current state.

## onSettingReady

### Syntax

**onSettingReady***(event)*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when setting the user's state to ready.

## onSettingWorkNotReady

### Syntax

**onSettingWorkNotReady***(event)*

### Description

Fires when setting the user's state to work not ready.

## onSettingWorkReady

### Syntax

**onSettingWorkReady***(event)*

### Description

Fires when setting the user's state to work ready.

## onStat

### Syntax

**onStat***(event)*

### Description

Fires when statistics are received.

## onStat1

### Syntax

**onStat1***(event)*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when statistics1 is received.

## onStat2

### Syntax

**onStat2***(event)*

### Description

Fires when statistics2 received.

## onUnknown

### Syntax

**onUnknown***(event)*

### Description

Fires when the user's state changes to unknown.

## onUnregistered

### Syntax

**onUnregistered***(event)*

### Description

Fires when the address gets unregistered.

## onUnregistering

### Syntax

**onUnregistering***(event)*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Description

Fires when unregistering the address.

## onWorkNotReady

### Syntax

**onWorkNotReady***(event)*

### Description

Fires when the user's state changes to work not ready.

## onWorkReady

### Syntax

**onWorkReady***(event)*

### Description

Fires when the user's state changes to work ready.

# UserCaps Class Constructor

The following is the UserCaps class constructor.

## UserCaps

### Syntax

**UserCaps***(strCaps)*

### Description

The UserCaps object describes the user capabilities.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *strCaps* | The user capabilities. |

### Returns

A UserCaps object.

## UserCaps Class Fields

The following are the UserCaps class fields

## canHandleChat

### Description

Chat task handling capability of the user.

Type: boolean.

## canHandleEmail

### Description

Email task handling capability of the user.

Type: boolean.

## canHandleGeneric

### Description

Generic task handling capability of the user.

Type: boolean.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## canHandleVoice

### Description

Voice task handling capability of the user.

Type: boolean.

## canLogin

### Description

Login capability of the user.

Type: boolean.

## canLogout

### Description

Logout capability of the user.

Type: boolean.

## canRefreshState

### Description

The refreshState capability of the user.

Type: boolean.

## canSetNotReady

### Description

The setNotReady capability of the user.

Type: boolean.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## canSetPresence

### Description

The setPresence capability of the user.

Type: boolean.

## canSetReady

### Description

The setReady capability of the user.

Type: boolean.

## canSetWorkNotReady

### Description

The setWorkNotReady capability of the user.

Type: boolean.

## canSetWorkReady

### Description

The setWorkReady capability of the user.

Type: boolean.

# UserData Class Constructor

The following is the UserData class constructor.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## UserData

### Syntax

```
UserData()
```

### Description

The UserData object describes the key-value pairs of attached user data.

### Parameters

None.

### Returns

A UserData object.

# UserData Class Fields

The following are the UserData class fields constants.

## UserData Class Field Constants

### Description

UserData class uses the following constants.

| *Value* | *Description* |
|---|---|
| COMPONENT | The component. |
| DESCR | The description |
| ICSCRIPTPROGRAMNAME | The IC script program name. |
| ICTYPE | The IC type. |
| MARKET | The market. |
| MENU | The menu. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Value* | *Description* |
|---|---|
| REFERENCEID | The reference ID |
| TARGET | The target. |
| URLABS | The absolute URL. |
| URLREL | The relative URL. |

# UserData Class Method

The following is the UserData class method.

## addKeyValue

### Syntax

**addKeyValue***(key,value)*

### Description

Add the key-value pair to the UserData object.

### Parameters

| *Parameter* | *Description* |
|---|---|
| *key* | The key. |
| *value* | The value. |

### Returns

None.

# UserStatistics1 Class Constructor

The following is the UseStatistics1 class constructor.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## UserStatistics1

### Syntax

```
UserStatistics1()
```

### Description

UQ User statistics sent on user refresh 1.

### Parameters

None.

### Returns

A UserStatistics1 object.

# UserStatistics1 Class Fields

The following are the UserStatistics1 class fields.

## availableCost

### Description

The amount of unused work.

## ctiBusy

### Description

Flag indicating that a user is engaged on CTI task.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## currentQueue

### Description

Current queue/group the User last logged into.

Type: string.

## mostRecentTaskData

### Description

Most recent task's data.

Type: string.

## mostRecentTaskId

### Description

Most recent task's ID.

Type: string.

## numTaskAccepted

### Description

The number of tasks accepted by the user.

Type: string.

## numTasksDone

### Description

The number of tasks done by the user.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## numTasksUnassigned

### Description

The number of tasks unassigned by the user.

Type: string.

## presenceText

### Description

The presence text for the user.

Type: string.

## reasonFlag

### Description

The reason flag for this event.

Type: string.

## state

### Description

Current state of the user.

Type: string.

## timeInCurrentState

### Description

Time in current state for the user.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### timeSinceLoggedIn

#### Description

The time since the user logged in.

Type: string.

## UserStatistics2 Class Constructor

The following is the UserStatistics2 class constructor.

### UserStatistics2

#### Syntax

```
UserStatistics2()
```

#### Description

UQ User statistics sent on user refresh 2 event.

#### Parameters

None.

#### Returns

A UserStatistics2 object.

## UserStatistics2 Class Fields

The following are the UserStatistics2 class fields.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## currentQueue

### Description

The current queue or group for the user.

Type: string.

## timeIdle

### Description

Idle time for the user.

Type: string.

## timeInCurrentState

### Description

The time in the current state for the user.

Type: string.

## timeNotReady

### Description

The time during which the user was in the not ready state.

Type: string.

## timeSinceLogin

### Description

The time since the user logged in.

Type: string.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## totalTimeAvailable

### Description

The total time during which the user has been in the available state.

Type: string.

## totalTimeUnavailable

### Description

The total time during which the user has been in the unavailable state.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 14

# Configuring the Email Channel

This chapter provides an overview of the email channel and discusses how to:

- Configure PeopleSoft Integration Broker for the email channel.
- Enable virus scanning .
- Demonstrate the email channel.

## Understanding the Email Channel

Because the email channel uses a PeopleSoft Integration Broker gateway, the email channel requires additional configuration outside of PeopleSoft MultiChannel Framework configuration pages.

PeopleSoft MultiChannel Framework (MCF) provides a framework for:

- Fetching emails from a mail server.
- Storing and retrieving email parts in a database.
- Managing attachments.
- Queueing of emails by the queue server.

Application developers use the PeopleCode application package PT_MCF_MAIL to develop PeopleSoft Application Engine programs and PeopleSoft Pure Internet Architecture components for email processing.

See *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Mail Classes."

PeopleSoft MultiChannel Framework email channel features include:

- The GETMAILTARGET target connector.
- A PeopleCode application package, PT_MCF_MAIL, to retrieve, store, and delete email.
- Support for both Post Office Protocol 3 (POP3) and Internet Message Access Protocol 4 (IMAP4) email protocols.
- Support for SSL connections.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Support for email attachments, including:
  - An attachments repository running on the same web server as PeopleSoft Integration Broker.
  - URL access to email attachments.
  - A relative addressing scheme to enable flexibility of repository location.
  - User- and role-based security to control access to attachments.
- The option to access email headers, attachments, and body from the mail server.
- Support for multiple message size thresholds to control distribution of email between the database, the attachment repository, and the mail server.
- Support for UTF8 Unicode as supported by the Sun Java Runtime Environment (JRE).

  See http://java.sun.com.
- Time zone conversions to support global service-level agreements.
- Sample email application pages demonstrating the functionality of the PT_MCF_MAIL application package.
- Ability to enable logging for MCFOutbound email by setting SMTP trace in psappsrv.cfg. The log file is created in the log directory defined in psappsrv.cfg.

  See *PeopleTools 8.51 PeopleBook: System and Server Administration*, "Setting Application Server Domain Parameters," SMTPTrace.

## Handling Email

This section describes some factors to consider when handling email using PeopleSoft MultiChannel Framework.

### *POP3 Versus IMAP4*

Most mail servers support simultaneous client connections through both POP3 and IMAP4. Using IMAP4 for your MCF email connection provides significant benefit over using POP3. IMAP4 allows for the use of email folders, which allows malformed emails to be set aside for separate processing. Quarantining malformed emails allows system administrators to remove or correct invalid emails without interrupting normal processing.

---

**Note.** Using IMAP4 for your MCF email connection does not preclude the use of POP3 for any other client connections to your mail server.

---

### *Time Zone Offsets*

Set the values of the email sent time zone offset (in minutes) and the receive time zone offset (in minutes) whenever possible. The default value is 800, which indicates time zone information is not available. When available, the values range from +720 to −720.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Connector Determination of Email Size

The connector sometimes cannot determine the size of the message. In such cases the size is set to 0 and an error message is written to the gateway error log.

### Malformed Emails

Emails with syntax that does not comply to the latest standard definitions may not be processed successfully by MCF. MCF will return as much as possible of noncompliant emails, however, at times, invalid parts will be ignored. For example, email addresses with invalid characters will not be displayed, such as two @ symbols. Also, if the header information is not valid, such as a malformed content type, the associated email part may not appear.

A common example of a malformed email is one that contains 8-bit encoded information in a header, such as the To, Cc or Subject field, or in an attachment file name. According to the standard definitions, this information must be encoded using 7-bit encoding, as described in RFC 2045 and others. This encoding is usually done by the email sender.

---

**Note.** The PeopleSoft email reader only supports email messages that follow the SMTP specification. Many email clients and mail servers have their own proprietary formats; therefore, be aware that PeopleSoft MCF will interpret these emails according to the RFC specification. For example, some mail servers and clients allow 8-bit encoded data in email headers, which the PeopleSoft email reader does not allow.

---

### Domain Validation

Application developers can use the method IsDomainNameValid to validate email addresses.

---

**Note.** The number of retries for domain validation is set by the parameter SMTPDNSTimeoutRetries in psappsrv.cfg.

---

See *PeopleTools 8.51 PeopleBook: System and Server Administration*, "Setting Application Server Domain Parameters," SMTP Settings and *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Mail Classes," IsDomainNameValid.

### Character Sets

If you wish to attach files which are named using characters outside of the character set recognized by your application server or process scheduler operating system, you should update the locale or regional settings of your operating system to allow those characters to be recognized.

### SSL Connections

Certificates are an important component in SSL communication for authentication of any party involved. In this case one party is Email Server and another party is PeopleSoft Applications. For the communication to work, you must import CA signer of the Server Certificate installed in the Email Server to pskey peoplesoft keystore stored in the web server. The path to pskey peoplesoft keystore is defined in secureFileKeystorePath property for the Gateway Properties.

See *PeopleTools 8.51 PeopleBook: System and Server Administration*, "Working with Oracle WebLogic," Importing Keys and Certificates Into the Keystore.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Connect data for outgoing emails can be configured in psappsrv.cfg or provided by the application using application package PT_MCF_MAIL.

See *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Mail Classes."

### *Virus Scanning*

You can enable virus scanning for inbound MCF Email attachments and outbound file attachments.

# Configuring PeopleSoft Integration Broker for the Email Channel

This section discusses how to:

- Configure the gateway.
- Configure GETMAILTARGET properties.

Configure nodes and gateways for the email channel in PeopleSoft Integration Broker.

## Pages Used to Configure Integration Broker for Email

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Node Info | IB_NODE | PeopleTools, Integration Broker, Node Definitions, Node Info | Define the MCF_GetMail node. |
| Connectors | IB_NODECONN | PeopleTools, Integration Broker, Node Definitions, Connectors | Define the MCF_GetMail connector. |
| Gateways | IB_GATEWAY | PeopleTools, Integration Broker, Gateways, Gateway ID | Set up the PeopleSoft Integration Broker gateway for the MCF_GetMail connector node. |

### *See Also*

*PeopleTools 8.51 PeopleBook: PeopleSoft Integration Broker Administration*, "Managing Integration Gateways"

## Configuring the Gateway

Access the Gateways page using the following navigation path:

PeopleTools, Integration Broker, Gateways, Gateway ID

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

1. Create a local gateway.
2. Enter the gateway URL in the form http://<gatewayservername>/PSIGW/PeopleSoftListeningConnector, where <gatewayservername> is the fully qualified host name for the gateway machine.

   For example, http://host1.peoplesoft.com/PSIGW/PeopleSoftListeningConnector.
3. Save the page.
4. Click Load Gateway Connectors.

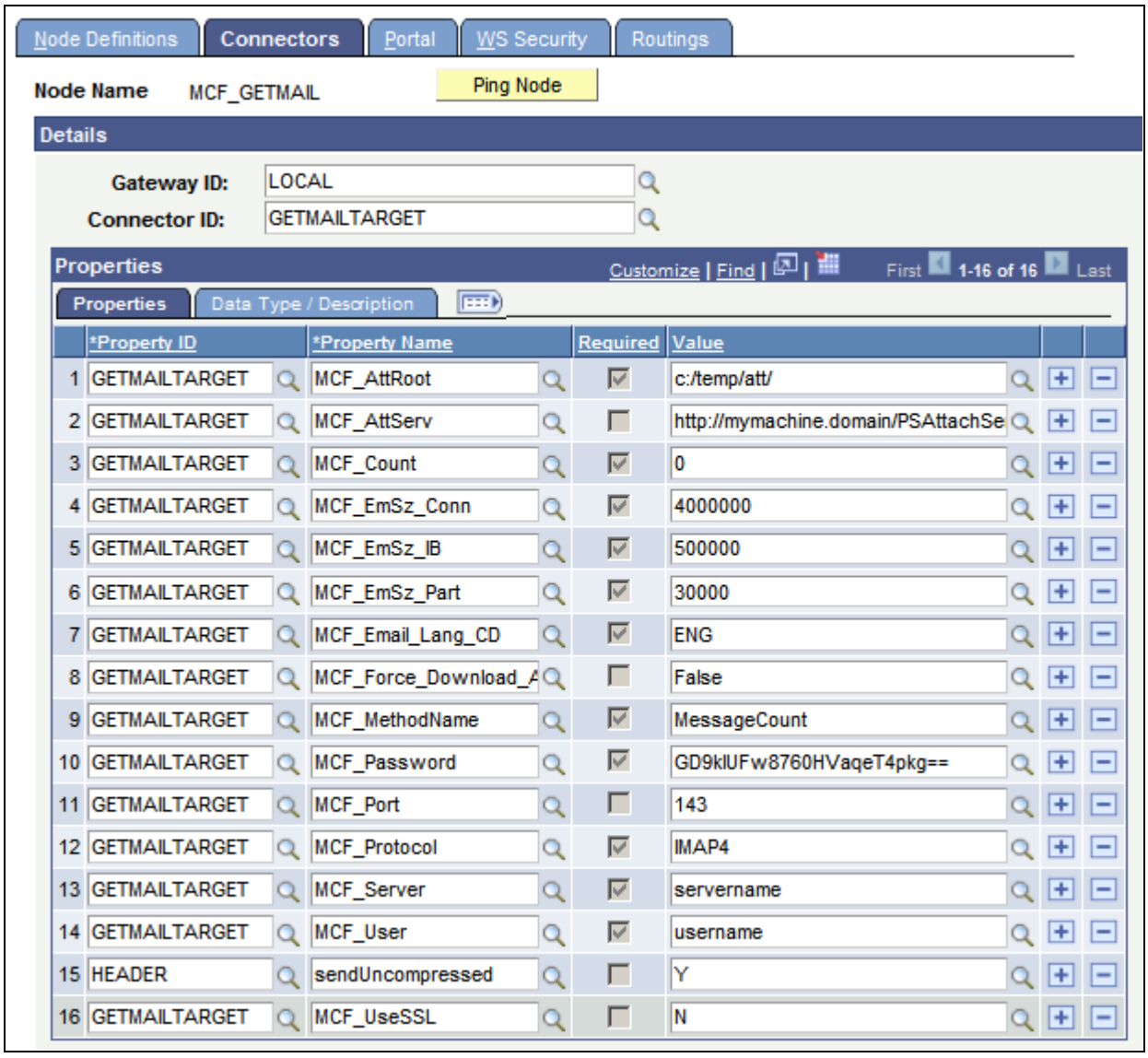
   The connector properties appear.
5. Save the page again.
6. Configure the GETMAILTARGET connector properties.

## Configuring GETMAILTARGET Properties

Access the Connector Properties page using the following navigation path:

PeopleTools, Integration Broker, Node Definitions, Connectors

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Connectors page

Configure GETMAILTARGET connector properties on the MCF_GETMAIL node. You can also set threshold values for email routed to the attachment repository.

The following table lists node properties:


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Property Name* | *Default Value* | *Description* |
| --- | --- | --- |
| MCF_AttRoot | *c:\temp\att* | Defines the location of the attachment repository on the gateway.<br><br>For example, if the attachment root is c:\temp\att, then attachments for emails addressed to user name *support* are downloaded to c:\temp\att\<mail server host name>\<mailbox>.<br><br>**Note.** The directory on the remote machine where the attachments are stored should be shared. The user permissions for should be stored on the node (local node) that will retrieve the attachments.<br><br>**Note.** The user name or mailbox portion of the attachment root path may include encoded characters.<br><br>All email attachments, and email parts not of the content types specified for the MCF_ContentTypes property, are always written to the attachment repository. A relative URL is returned to the application that allows the application to access the attachment.<br><br>The attachment retrieval depends on the file system of the operating system. For Windows NTFS/FAT 16 file system, the folder length or maximum characters of the file extension is 256/512. For Unix and Solaris UFS file system, the length is 256.<br><br>Attachment security is managed at a user and role level using the PT_MCF_EMAIL application package.<br><br>Users can configure attachment folder paths using metastrings to enable multiple folders based on several variables:<br><br>• *%OPRID%:* PeopleSoft user ID.<br>• *%DBNAME%:* database name.<br>• *%CURRDATE%:* current date.<br>• *%CURRHOUR%:* current hour. |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Property Name | Default Value | Description |
|---|---|---|
| MCF_AttServ | *http://<machine_name>.<domain_name>/PSAttachServlet/ps/* | Defines the location of the MCF attachment repository servlet, located on the gateway web server.<br>Attachment relative URLs are appended to this address to create a fully qualified URL to reference an attachment in the repository.<br>**Note.** When you are setting a remote gateway to access email attachments stored on the remote machine, the database that the gateway/psattachservlet accesses should have the permissions to view the attachments.<br>**Note.** If your web server is installed with a PeopleSoft Pure Internet Architecture site name other than *ps*, include the site name in the specified path. |
| MCF_ClientCertAlias | ClientCert | Provide the alias of the certificate in pskey.<br>**Note.** The client certificate must be imported in pskey.<br>See *PeopleTools 8.51 PeopleBook: PeopleSoft Integration Broker Administration*, "Setting Up Secure Integration Environments," Generating Private and Public Key Pairs. |
| MCF_ClientCertPass | password | Enter the password for the certificate.<br>**Note.** The alias and the password must match the alias and password of the imported certificate in pskey keystore. |
| MCF_ContentTypes | *text/plain* | Specifies email content types that will be stored in the database. Content types not specified will be stored as attachments. Enter content types separated by commas.<br>For example, you can Enter *text/html* and *text/xml* to enable storage in the database of HTML email and email containing XML.<br>The content type *text/plain* is always stored in the database, regardless of values specified for MCF_ContentTypes.<br>Do not include binary encoded content types in the content types list unless the application can handle base64 encoded text. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Property Name | Default Value | Description |
|---|---|---|
| MCF_Count | *0* | Defines the default number of emails retrieved from the mail server unless otherwise specified in the SyncRequest.<br>The PeopleSoft MultiChannel Framework email application package always specifies the number of emails to be retrieved and does not reference this property. |
| MCF_EmSz_Conn | *4000000* | Defines the connector threshold, in bytes.<br>This parameter ensures that emails retrieved from a POP3 mail server do not exceed the available memory on the gateway server. Email content retrieved from IMAP4 servers can be streamed to a file on the attachment repository, but content retrieved from POP3 servers must first be read into memory before being written to a file. Therefore, the MCF_EmSz_Conn threshold can be set very high for IMAP4 mail servers.<br>PeopleSoft MultiChannel Framework does not process an email if its size is greater than this value. The status returned to the application is *1*. The triggering email is neither downloaded nor deleted from the mail server. Only the email header is returned to the application.<br>See *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Mail Classes." |
| MCF_EmSz_IB | *500000* | Defines the PeopleSoft Integration Broker threshold, in bytes.<br>This is the maximum size message that the GETMAILTARGET connector can send through PeopleSoft Integration Broker. The purpose of this threshold is to ensure that retrieved emails do not exceed the available memory on the gateway server and that the server running the application that requested the emails.<br>A PeopleSoft Integration Broker message can contain one or more emails, depending on the number of emails the application retrieves per message. As soon as this threshold is reached, the remainder of the triggering email is written to the repository, and no more emails are retrieved. Emails fetched prior to the triggering email are returned normally. The triggering email is returned with a return status of *2*, indicating that some of its parts were written to the repository.<br>See *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Mail Classes." |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Property Name | Default Value | Description |
|---|---|---|
| MCF_EmSz_Part | *30000* | Defines the text part threshold, in bytes.<br>If any text part of an email exceeds this value , the part is routed to the attachment repository. This ensures that excessively large objects of text are not written to the database.<br>For databases with limitations on the maximum size of rows or long text field lengths, this value must be lower than that limit to avoid SQL errors when saving emails. |
| MCF_Email_Lang_CD | *ENG* | Defines the expected language of emails downloaded by this node.<br>This code is included in the email header returned to the application. Configure a node for each language you expect to handle, because Simple Mail Transfer Protocol (SMTP) mail servers and clients do not guarantee correct identification of the email's language when creating an email. |
| MCF_FetchSize | *819200* | Determines the number of bytes retrieved for an IMAP email in a single fetch. It also controls the size of the buffer ( in bytes) used to write email attachments to the repository.<br>Setting MCF_FetchSize to a higher number uses more memory, but it may also increase the speed of download for larger emails and attachments. |
| MCF_Force_Download_Attachments | *False* | Enables downloading attachments that might otherwise be interpreted as text. If set to True all attachments, including text/plain, irrespective of their size, are downloaded to the attachment repository. This property enables reading of non-ASCII attachments. |
| MCF_MethodName | *MessageCount* | Defines the methods associated with the application class package.<br>The defined method is the default method used unless otherwise specified in the SyncRequest. The email application package always specifies the method to be used and does not reference this property.<br>See *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Mail Classes." |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Property Name* | *Default Value* | *Description* |
|---|---|---|
| MCF_Password | None | Defines the default password for the mailbox unless otherwise specified in the SyncRequest.<br>See *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Mail Classes." |
| MCF_Port | *143* | Defines the mail server port to be used.<br>By default, POP3 servers use port 110 and IMAP4 servers use port 143. Confirm the port number with your system administrator and set it accordingly here. |
| MCF_Protocol | *IMAP4* | Defines the protocol used by the connector to access emails on the mail server.<br>Supported values are *POP3* or *IMAP4.* |
| MCF_Quarantine | *Quarantine* | Defines a quarantine folder for email meeting the following criteria:<br>• Connector size overflow.<br>• Unsupported encoding.<br>• Unknown Java exception.<br>• JavaMail content error.<br>• No attachment repository.<br>• Mail parse exception.<br>The quarantine folder is created for each user account and must be managed by each email user, not by system administrators.<br>**Note.** Folders are supported only on IMAP4 mail servers. |
| MCF_Server | None | Defines the fully qualified host name of the mail server.<br>This is the default fully qualified host name of the mail server, unless otherwise specified in the SyncRequest. For example, bigserver.peoplesoft.com.<br>See *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Mail Classes." |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Property Name* | *Default Value* | *Description* |
|---|---|---|
| MCF_User | None | Defines the user name for the email account (mailbox) being accessed.<br>This is the default user name used unless otherwise specified in the SyncRequest.<br>For example, if emails are addressed to support@peoplesoft.com, the user name is *support*. |
| MCF_UseSSL | *N* | Indicates if an SSL connection will be attempted. If set to *Y*, an SSL connection will be attempted. MCF_Port and MCF_Protocol will be used as port and protocol for making the SSL connection.<br>A value of *N* indicates a Non SSL connection. |

**Note.** The POP3 and IMAP4 email protocols calculate message size differently. POP3 does not include header size, but IMAP4 does. As a result, message sizes affecting thresholds behave differently depending on the protocol used.

Multipurpose Internet Mail Extensions (MIME), is an Internet standard for representing multipart and multimedia data in email so that they can be exchanged between different email systems. The parts may be nested. PeopleSoft embeds the Sun JavaMail API to implement support for the MIME standard. However, all parts of an email are represented in PeopleSoft as level 1 rowsets, regardless of whatever hierarchy existed in the original email. Email clients determine the MIME format of the emails sent from them, so identical email content sent from different email clients may have a different MIME structures.

# Enabling Virus Scanning

This section discusses how:

- Locate virusscan.xml file for your web server.
- Configure the virusscan.xml file.
- Use virus scan error logs.

## Locating virusscan.xml File for Your Web Server

Virus scanning can be enabled for inbound MCF Email and AddAttachment, by configuring the virusscan.xml file for your virus scan engine. The location of this file depends on your web server:

- WebLogic:

  <PIA_HOME>/webserv/<domain>/applications/peoplesoft/PSIGW.war/WEB-INF/classes/psft/pt8/virusscan

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- Websphere:

  <PIA_HOME>/webserv/<domain>/installedApps/<domain>NodeCell/<domain>.ear/PSIGW.war/ WEB-INF/classes/psft/pt8/virusscan

See *PeopleTools 8.51 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions," Continue and *PeopleTools 8.51 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions," ConvertChar.

## Configuring the virusscan.xml File

To enable the virus scanning feature:

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

1. Open VirusScan.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<Providers disableAll="True" logFile="./servers/PIA/logs/VirusScan%u.log">
    <!-- Sample Configuration for Symantec Engine
 <Provider>
        <name>Symantec</name>
      <class>psft.pt8.virusscan.provider.GenericVirusScanProviderImpl</class>
      <icapversion>ICAP/1.0</icapversion>
         <service-name>/SYMCScanResp-AV</service-name>
         <policycommand>?action=SCAN</policycommand>
         <address>152.68.144.44</address>
         <port>1344</port>
         <disable>false</disable>
    </Provider>-->

    <!-- Configure your own proivider -->
    <Provider>
    <!-- Provider Name of the Scan Engine -->
    <name></name>
    <!-- Provider Class of the Scan Engine.
         psft.pt8.virusscan.provider.GenericVirusScanProviderImpl is
         the default
         provider class.  -->
    <class>psft.pt8.virusscan.provider.GenericVirusScanProviderImpl</class>
    <!-- ICAP version -->
    <icapversion>ICAP/1.0</icapversion>
    <!-- ICAP ServiceName. The Service Name changes from Scan Engine to
   Scan Engine.
         This is the name Scan Engine Service is will be hosted with -->
    <service-name></service-name>
    <!-- RESPMOD extra commands, These are the RESPMOD commands
   (SEE ICAP Protocol).
         Usually these commands will be chainging from Engine to Engine
    -->
    <policycommand></policycommand>
    <!-- IP Address of Scan Engine host> -->
    <address></address>
    <!-- IP Port of Scan Engine host -->
    <port></port>
    <!-- Disable scanning for this provider -->
    <disable></disable>
    <!--
         Default codes = 200 and 204 for clean, 201,403 for infected
         Use these tags to change the behaivior if needed
         <clean>200,204</clean>
         <infected>201,403</infected>
         -->
</Provider>
</Providers>
```

---

**Note.** A sample configuration for Symantec Engine is provided in the remarks.

---

2. In the Providers tag, set the attribute *disableAll* to False .

```
<Providers disableAll="False" logFile="./servers/PIA/logs/VirusScan%u.log">
```


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

3. Multiple scan engine can be configured under <Providers>. Each <Provider> tag represents one scan engine. All configured scan engines will check for viruses. For each <Provider> tag enter values for the tags:

| *Tag* | *Description* | *Example Value for Scan Engine* |
|---|---|---|
| <name> | Provider Name of the Scan Engine | Symantec |
| <class> | Provider Class of the Scan Engine<br>Default provider class is:<br>`psft.pt8.virusscan.provider.⇒ GenericVirusScanProviderImpl` | `psft.pt8.virusscan.provider.⇒ GenericVirusScanProviderImpl` |
| <icapversion> | ICAP version | ICAP/1.0 |
| <service-name> | Service name for the scan engine host. | /SYMCScanResp-AV |
| <policycommand> | Policy command used by the Scan Engine. Only SCAN is supported. | ?action=SCAN |
| <address> | IP address of Scan Engine host. | IP address of the machine where the scan engine is running |
| <port> | IP port of Scan Engine host. | Port where the scan engine is running |
| <disable> | Disable scanning for this provider. | false |
| <clean> | Default codes = 200 and 204 for clean.<br>You can use this tag to change the behavior if needed. | 200,204 |
| <infected> | Default codes = 201 and 403 for infected<br>You can use this tag to change the behavior if needed. | 201,403 |

## Using Virus Scan Error Logs

There are two type of logs generated virus scanning logs and error logs.

### *Virus Scanning Logs*

Virus Scanning logs are the only interface with the scanning engine. These logs are located in the path indicated by the *logfile* property in VirusScanning.xml. If there is a failure, the details will be logged *ig.errorLog.filename* in integrationGateway.properties

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Error Logs

If there is a failure, the details will be logged *ig.errorLog.filename* in integrationGateway.properties For example, ig.errorLog.filename in <PIA_HOME>/webserv/<domain>/applications /peoplesoft/PSIGW.war/WEB-INF/integrationGateway.properties.

The return value when the virus scan for mail attachments is REPOSITORY_FAILURE = 8.

See *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "Mail Classes," Error Messages Returned by MCFGetMail Class Methods.

If there are any errors during file processing the error codes listed in this table will be generated.

| Error Code | Description |
|---|---|
| %Attachment_ViolationFound | File violation detected by Virus scan engine. |
| %Attachment_VirusScanError | Virus scan engine error. |
| %Attachment_VirusConfigError | Virus scan engine configuration error. |
| %Attachment_VirusConnectError | Virus Scan engine connection error. |

## Demonstrating the Email Channel

To demonstrate the email channel, use the Email Sample Pages (MCFEM_DEMOERMS_CMP) component.

This section discusses how to:

- Use the GetMail - Server page.
- Use the MailStore - DB page.

The PeopleSoft system provides email sample pages to demonstrate the functionality of the PT_MCF_MAIL application package.

**Note.** The email sample pages are not intended for any purpose other than demonstration and troubleshooting. They should not be used in production.

The email sample pages can be also used to test the configuration of your integration gateway and MCF_GETMAIL node.

**Note.** Using the email sample pages requires access to a mail server.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

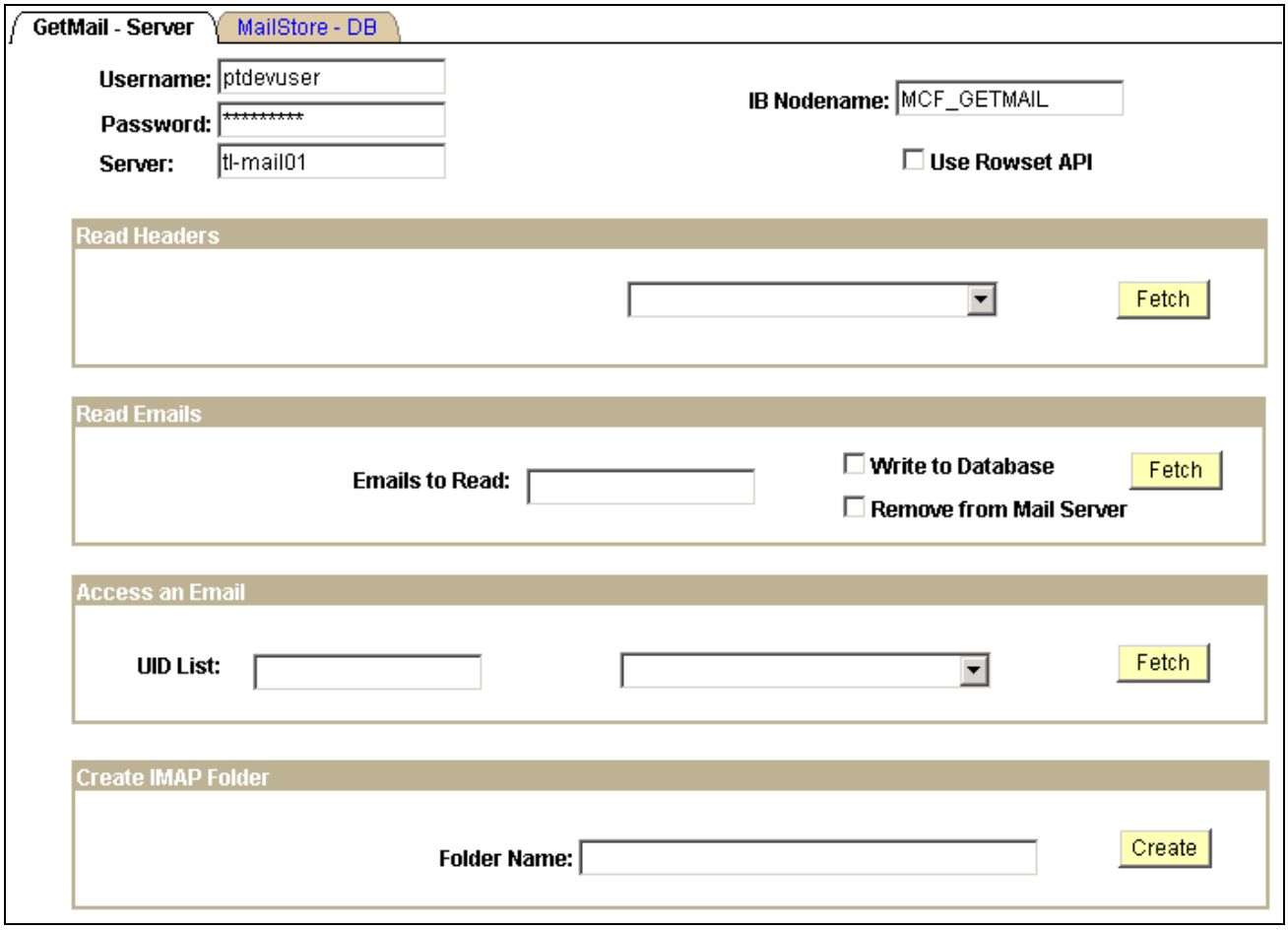
## Pages Used to Demonstrate the Email Channel

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| GetMail - Server | MCFGETMAIL_PG | PeopleTools, MultiChannel Framework, Email, Sample Pages, GetMail - Server | Demonstrate GetMail functionality. |
| MailStore - DB | MCFMAILSTORE_PG | PeopleTools, MultiChannel Framework, Email, Samples Pages, MailStore - DB | Demonstrate MailStore functionality. |

## Using the GetMail - Server Page

Access the GetMail - Server page using the following navigation path:

PeopleTools, MultiChannel Framework, Email, Sample Pages, GetMail - Server




The GetMail - Server page having the following editable fields: User Name, Password, IB Nodename, Use Rowset API, Read Headers, Emails to Read, Write to Database, Remove from Mail Server, UID List, and IMAP Folder Name.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

This page demonstrates reading or deleting emails from the mail server. It also demonstrates the storage and retrieval of emails to and from the database tables. Output of the response to each of the requests is written to a file at %PS_SERVDIR%/files/mcfdata.out on the application server. Check this file after each test for the output data.

To demonstrate email functionality, send an email to the user name and server that you enter on the GetMail - Server page.

**Username** and **Password** Enter a valid username and password for an account on the mail server.

**Server** Enter the mail server.

**IB Nodename** (Integration Broker node name) Enter the PeopleSoft Integration Broker node used to access email from the mail server.

The default is MCF_GETMAIL

**Use Rowset API** Select to use the rowset-based GetMail API.

Deselect to use the new Mail Classes API.

### *Read Headers*

Select a method to be called from the drop-down list box, then click Fetch. Select from:

***Message Count:*** Writes the number of emails in the specified mailbox to the output file mcfdata.out.

***ReadHeadersWithAttach:*** Writes headers and attachment information for all emails in the specified mailbox to the output file mcfdata.out.

### *Read Emails*

Enter the number of emails to read, select other parameters, then click Fetch to read emails.

**Emails to Read** Enter the number of emails to retrieve from the specified mailbox and write to the output file mcfdata.out.

**Write to Database** Select to have the retrieved emails written to the database as well as to the output file mcfdata.out.

**Remove from Mail Server** Select to delete emails from the mail server after they have been retrieved .

### *Access an Email*

Enter an email UID list, select a method from the drop-down list box, and then click Fetch to access the selected emails.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**UID List** (unique identifier list) — Enter the unique ID of an email to be retrieved or deleted. To delete, you can enter a comma-delimited list of unique email IDs (UID) to be deleted.

Each email has a unique number (the UID) associated with it that is permanently guaranteed not to refer to any other email in the mailbox. To find the UID of an email, run the *ReadHeadersWith Attach* option first, which returns the UID for each email. If an invalid UID is specified, one empty row is returned.

***Read Email w/ attachment*** — Retrieves the specified email from the specified mailbox and writes it to the output file mcfdata.out.

***Remove Message*** — Deletes the specified email from the specified mailbox.

### *Create IMAP Folder*

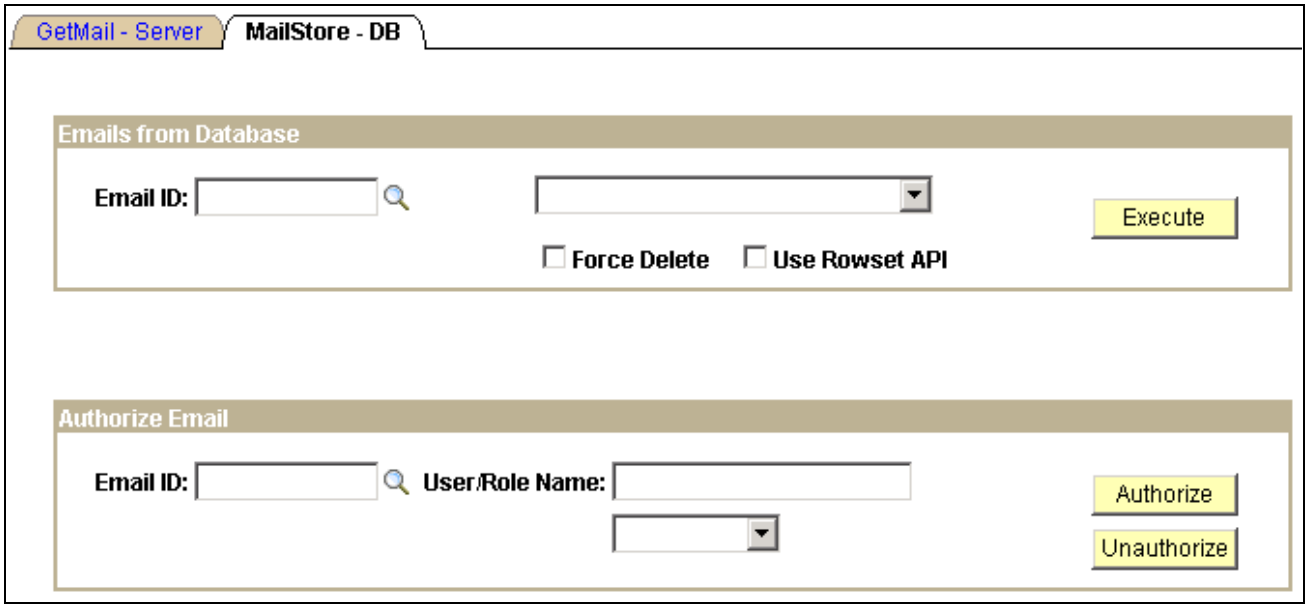
Create an email folder; only valid for IMAP4 mail servers.

**Folder Name** — Enter the name of the folder to create.

## Using the MailStore - DB Page

Access the MailStore - DB page using the following navigation path:

PeopleTools, MultiChannel Framework, Email, Samples Pages, MailStore - DB




The MailStore - DB page having the Emails from Database and Authorize Email group boxes.

View an example of how to access emails from the database (how to retrieve and delete email) and how to authorize email attachments for viewing or deletion using the PT_MCF_MAIL application package provided with PeopleSoft MCF.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

**Email ID** Enter the PeopleSoft email ID (not to be confused with the UID) with which the email was saved to the database.

### *Emails from Database*

Enter an email ID, select an action from the drop-down list box, and then click Execute to perform the selected action.

***Delete Email from Database*** Deletes the specified email from the database and any corresponding attachments from the repository.

***Retrieve Email*** Retrieves the specified email from the database and writes it to the output file mcfdata.out.

**Force Delete** Select after entering *Delete Email from Database* to force the deletion even if an error occurs when deleting associated attachments from the repository.

**Use Rowset API** Select to use the rowset-based GetMail API.

Deselect to use the new Mail Classes API.

### *Authorize Email*

Enter an email ID, a user role or name, and click Authorize or Unauthorize to perform the specified action.

**Email ID** Enter the PeopleSoft email ID (not to be confused with the UID) with which the email was saved to the database.

**User/Role Name** Enter a PeopleSoft user ID or role to be authorized or unauthorized to view the attachment associated with the selected email.

After entering an ID or role, select *User* or *Role,* as appropriate, from the drop-down list box.

### *See Also*

Chapter 11, "Using PeopleSoft MCF Broadcast and Working with Sample Pages," Using the Email Sample Page, page 176

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Chapter 15

# Configuring Instant Messaging in PeopleSoft MultiChannel Framework

This chapter provides an overview of instant messaging and discusses how to:

- Configure instant messaging.
- Use single button presence.
- Use the instant messaging sample pages.

***See Also***

*PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "MCFIMInfo Class"

## Understanding Instant Messaging

PeopleSoft MCF extends the HTML chat functionality delivered in PeopleTools 8.42 to popular instant messaging networks. Users can initiate an outbound chat with a customer using instant messaging networks such as IBM/Lotus Sametime, or Yahoo! Messenger. This functionality enables an external user to participate in a business transaction through instant messaging.

PeopleSoft MCF enables instant messaging (IM) through public networks (GTALK and Yahoo! Messenger), an enterprise Lotus Sametime Connect network or XMPP Instant Messaging server.

You can enable a user to initiate an instant messaging session from any page using either a button or PeopleCode.

The PeopleSoft system does not supply the client for any supported instant messaging software. Each user must install the appropriate client software. However, presence detection for GTALK and Yahoo! Messenger users functions even without installed client software.

**Note.** The instant messaging presence detection integration (the IM Integration) in PeopleTools is subject to the respective IM vendor's network availability. In addition, the IM Integration is based on protocols made available by the respective IM Network vendors. Such protocols are subject to change at any time by the vendor. Therefore, Oracle is not responsible for the availability or performance of any of the IM networks or for any change in an IM vendor's protocol that may render the IM Integration inoperable.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Configuring Instant Messaging Servers*

The supported public instant messaging networks, Yahoo! Messenger and GTALK, , require installation of client software, but no server configuration. Installation and configuration of the client software is the responsibility of your administrators and customers, and is beyond the scope of this document.

Lotus Sametime Connect is an instant messaging network running on an enterprise Sametime/Domino server, which must be configured to support PeopleSoft MultiChannel Framework instant messaging. To administer instant messaging with Sametime Connect, install Sametime Links and configure the server to enable anonymous users to connect to the server and query for the presence of other known users. Refer to your Lotus/Domino server documentation for more information.

For XMPP Instant Messaging servers, you will need to configure the XMPP domain in order to detect user presence.

# Configuring Instant Messaging

In MCF you can enable instant messaging for :

- GTALK
- SAMETIME
- XMPP
- YAHOO

This section discusses how to configure instant messaging.

## Page Used to Configure Instant Messaging

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Configuration | MCF_IM_CFG_PG | PeopleTools, MultiChannel Framework, Instant Messaging, Configuration | Enter or edit a server IP address for instant messaging domains. |
| Configuration | MCF_IM_SRVR_CFG_PG | PeopleTools, MultiChannel Framework, Instant Messaging, Presence, Servers Configuration | Enter or edit an XMPP IM domain. |
| XMPP Servers Configuration | MCF_XMPPSRVRS_CFG | Click the Configure XMPP Servers link from the Configuration page. | Configure XMPP domains. |
| General Profile Information | USER_SELF_SERVICE | My System Profile | Configure IM users for XMPP. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Configuring Instant Messaging

Select PeopleTools, MultiChannel Framework, Instant Messaging, Configuration.

Configuration page displaying the IM Domain for Yahoo

**IM Domain** The instant messaging service name.

**Server IP Address** The host name and port (if applicable) of the instant messaging service.

***Warning!*** Do not alter the server address for Yahoo! or GTALK .

The server addresses for Yahoo! and GTALK are preconfigured. However, you must enter the address for your Sametime Connect server.

**Enabled** Select Enabled to enable the configured instant messaging service.

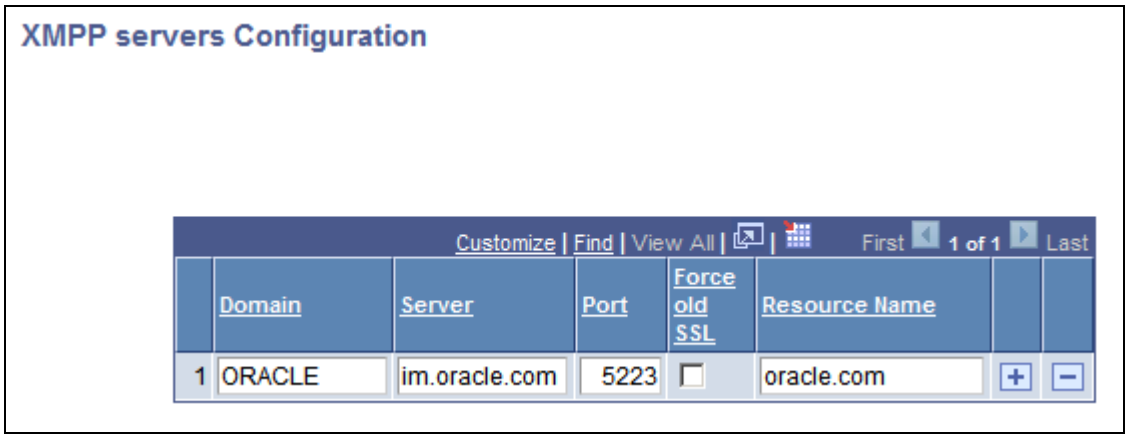
If the enabled flag is deselected, the network is still available, but the buttons will always show presence as offline.

## Configuring XMPP Domains

To configure XMPP servers:

1. Select PeopleTools, MultiChannel Framework, Instant Messaging, Configuration.
2. Click the Search button and select XMPP.
3. Click the Configure XMPP Servers link from the Configuration page.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




XMPP servers configuration page

Configure as many XMPP domains as required.

**Domain** Enter IM server domain.

**Server** Enter server information.

**Port** Enter the port for the IM server.

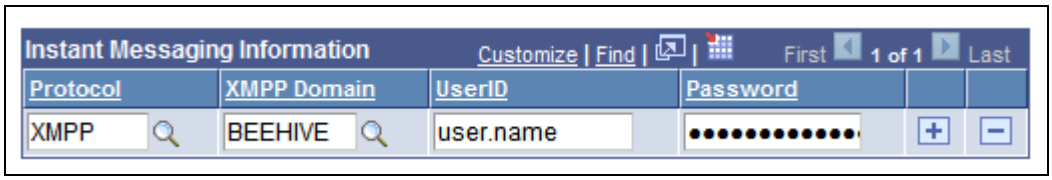
**Force old SSL** Select to use old SLL port 5223.

**Resource Name** Resource name for the domain name of the XMPP server. For example: oracle.com.

## Configuring IM Users for XMPP Servers

In order to obtain user presence information from XMPP configured IM servers, the user must login to the server. Each user needs to enter their userid and password for the XMPP server. If XMPP login information has been entered for a user profile, the login process to the configured XMPP server will happen automatically when the PeopleSoft user signs on to PeopleSoft. This login process takes place in the background and will not stop the user from performing other operations.

Select My System Profile or PeopleTools, Security, User Profile, User Profile and click the Instant Messaging Information link.




My System Profile-Instant Messaging Information

Enter your user ID and password information for each XMPP domain you are using.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Viewing IMServlet Logs

When troubleshooting or monitoring the IMServlet, you can view the IMServlet.log to gather information specific to the IMServlet. The log file is located in:

PS_HOME\webserv\peoplesoft\servers\PIA\logs

# Using Single Button Presence

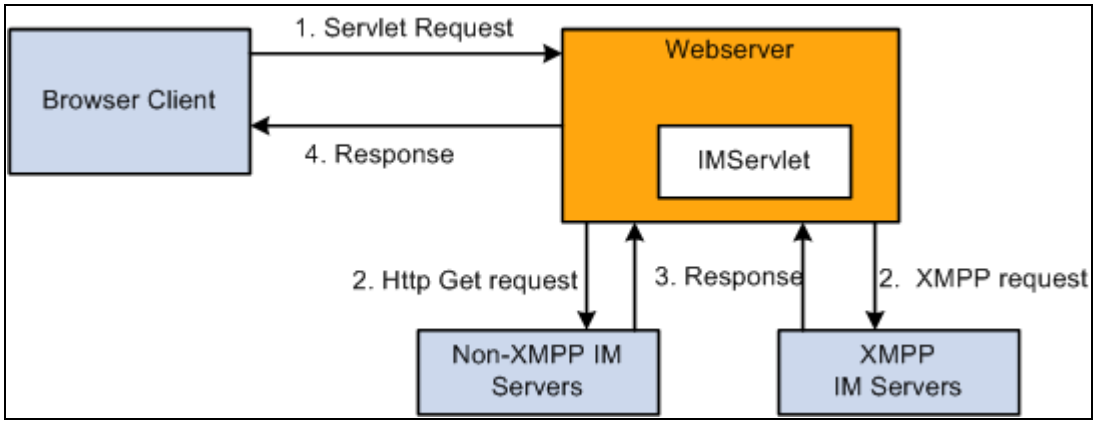
This section provides an overview of presence detection and discusses how to:

- Configure servers.
- Configure screen names.
- Test single presence button.
- Develop an application with instant messaging action button for presence detection.

## Understanding Presence Detection

The IM Servlet is implemented and hosted in the web server for the purpose of communicating with IM servers to get presence information. Therefore, browsers can send http Get request to IM Servlet and get presence information. The IM Servlet directly communicates with non-XMPP IM servers to fetch presence information and uses Smack API to communicate with XMPP servers. To improve the performance and response time the IM Servlet supports bulk requests for getting presence of multiple screen names using one request. In order to get the presence of a buddy from an XMPP server, the user needs to login into the XMPP server and maintain an active session. For non-XMPP servers the user does not need to login into IM server to fetch the presence information. This active session is maintained in the IM Servlet.

This diagram shows the instant messaging feature:




Instant messaging architecture

1. Browser sends get presence request to IM Servlet

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

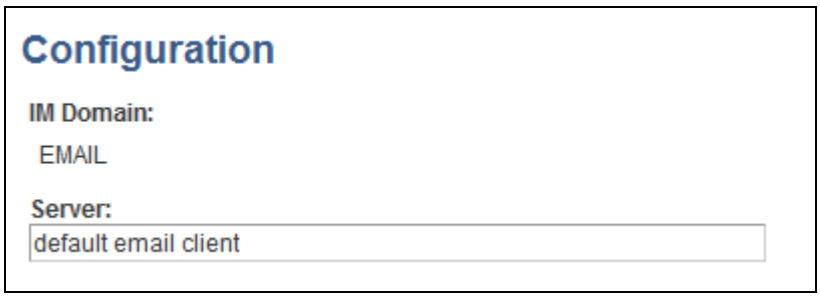
2. For non-XMPP servers, IM Servlet will send http get request to corresponding Instant Messaging server to find out presence information. For XMPP servers, IM Servlet will contact XMPP server via smack API to get presence information.
3. Presence information is returned to IM Servlet.
4. Presence is sent to the client browser.

## Pages Used to Configure Single Presence Button

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Configuration | MCF_IM_SRVR_CFG_PG | PeopleTools, MultiChannel Framework, Instant Messaging, Presence, Servers Configuration | Configure IM servers. |
| Screen names configuration | MCF_IMSCRNNAMES_PG | PeopleTools, MultiChannel Framework, Instant Messaging, Presence, Screen names configuration | Add and delete screen names for instant messaging. |
| Single button presence | MCF_PRS_DEMO_PG | PeopleTools, MultiChannel Framework, Instant Messaging, Presence, Sample page | Test single button presence. |

## Configuring Servers

PeopleTools, MultiChannel Framework, Instant Messaging, Presence, Servers Configuration.




Configuration page showing email example

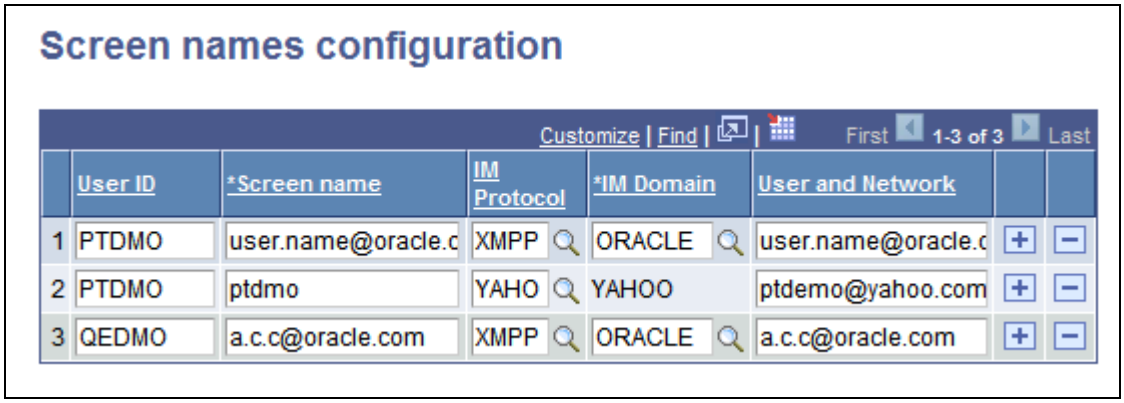
The configuration page is the same page used to configure instant messaging servers. When you select the page under the Presence menu, you can also configure Email, MSN and Skype.

See Chapter 15, "Configuring Instant Messaging in PeopleSoft MultiChannel Framework," Configuring Instant Messaging, page 523.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Configuring Screen Names

Select PeopleTools, MultiChannel Framework, Instant Messaging, Presence, Screen names configuration.




Screen names configuration page

**User ID** Enter PeopleSoft user ID.

**Screen name** ID used to obtain presence for this user.

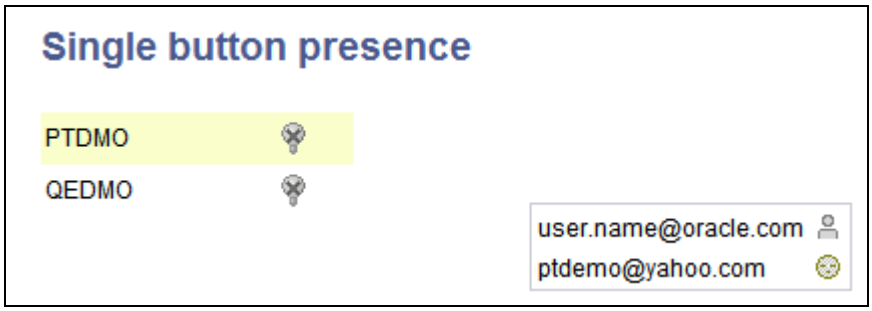
**IM Protocol** Select the protocol.

**IM Domain** Select the instant messaging domain.

**User and Network** The value entered here is used for display purposes in Single Button Presence.

## Testing Single Presence Button

Select PeopleTools, MultiChannel Framework, Instant Messaging, Presence, Sample page.




Single button presence page

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

The single presence icon represents a collective presence of all screen names. When a user moves the mouse pointer over a single presence icon, it will open a menu with all user screen names and their presence. If any of the screen names are online, you can click the online icon and it will launch the corresponding IM Messenger to send IM messages. When the user clicks on an email icon, it will launch default configured email client. When you click on Skype, it will launch Skype application and allow you to login to Skype.

## Developing an Application with Instant Messaging Action Button for Presence Detection

In order to create an application with instant messaging action button for presence detection:

1. Create a page and add a button.
2. Click on properties of a push button and choose *Instant Messaging Action* for the destination field.
3. Choose a record and field name where screen name is in the format for the instant messaging domain, as listed in this table:

| ***Instant Messaging Domain*** | ***Required format of screen name*** |
|---|---|
| YAHOO | screenname@YAHOO |
| XMPP | screenname @DOMAIN@XMPP |
| GTALK | screenname@GTALK |

4. Save page.

# Using the Instant Messaging Sample Pages

To demonstrate instant messaging, use the Instant Messaging Sample Pages (MCF_IM_DEMO_CMP) component.

This section discusses how to:

- Use the PeopleCode Sample page.
- Use the Button Sample page.

## Pages Used to Test Instant Messaging

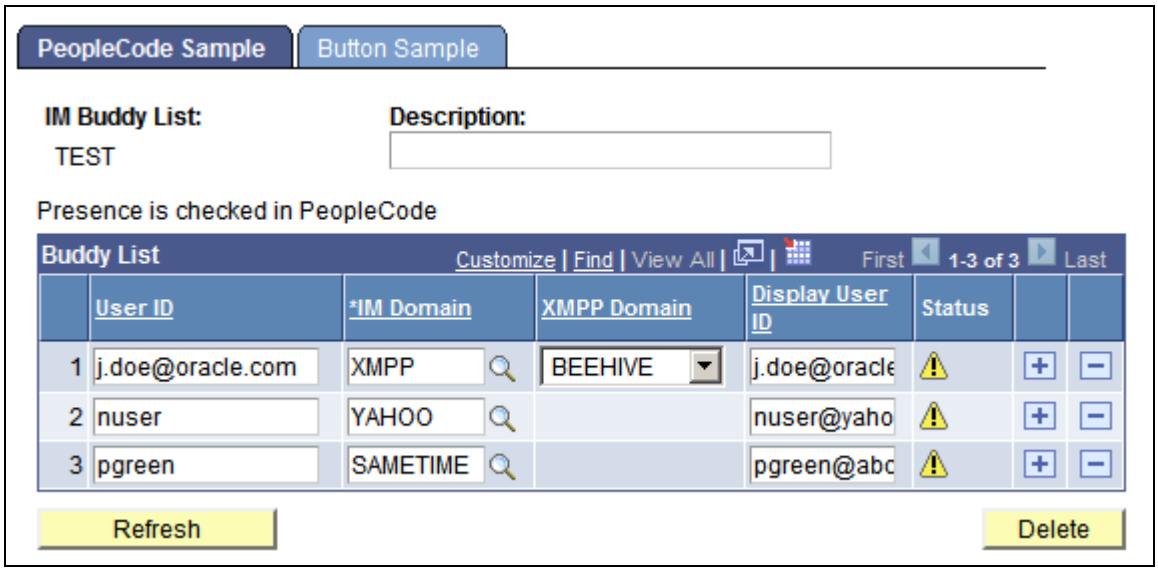
| ***Page Name*** | ***Definition Name*** | ***Navigation*** | ***Usage*** |
|---|---|---|---|
| PeopleCode Sample | MCF_IM_DEMO_PG | PeopleTools, MultiChannel Framework, Instant Messaging, Sample Pages, PeopleCode Sample | Set up buddy lists and view presence for configured instant messaging clients using PeopleCode. |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Page Name* | *Definition Name* | *Navigation* | *Usage* |
|---|---|---|---|
| Button Sample | MCF_IM_DEMO1_PG | PeopleTools, MultiChannel Framework, Instant Messaging, Sample Pages, Button Sample | View presence for configured instant messaging clients using the instant messaging button. |

## Using the PeopleCode Sample Page

Select PeopleTools, MultiChannel Framework, Instant Messaging, Sample Pages and add a new value.




PeopleCode Sample page

**Note.** The PeopleCode Sample does not support presence detection for GTALK and XMPP domains.

The PeopleCode Sample page demonstrates the use of instant messaging PeopleCode. The sample pages are intended for demonstration purposes and should not be used in production.

See *PeopleTools 8.51 PeopleBook: PeopleCode API Reference*, "MCFIMInfo Class."

**IM Buddy List** Displays an identifier for the list.

**Description** Enter a description for the list.

**User ID** Enter the user ID.

The user ID must be that of a valid user of the instant messaging service specified in the IM Domain field.

**IM Domain** Select the instant messaging service associated with the specified user ID from the drop-down list box.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| | |
|---|---|
| **XMPP Domain** | Select IM server domain. |
| **Display User Name** | Name used for display purpose. In GTalk, the actual user ID used for detecting presence could be up to 200 characters, this field is purpose of display. |
| **Status** | An icon displays the presence status of the specified user. The icons vary with the instant messaging service.<br>Click the icon to initiate an instant messaging session with the selected user. The user must be online.<br>**Note.** Yahoo! does not update presence automatically. Use the Refresh Presence button to check the current status of Yahoo! users. |
| **Refresh Presence** | Click the Refresh Presence button to check the status of Yahoo users.<br>**Note.** PeopleCode does not support presence detection in Sametime, so the Sametime user status does not appear on the PeopleCode Sample page. Use the instant messaging button if you intend to check presence of or connect to Sametime users. |

**Note.** If you get a warning symbol with the network, even though you have specified the correct address of IM server, provide values for Proxy host and Proxy port in the pasappsrv.cfg file.

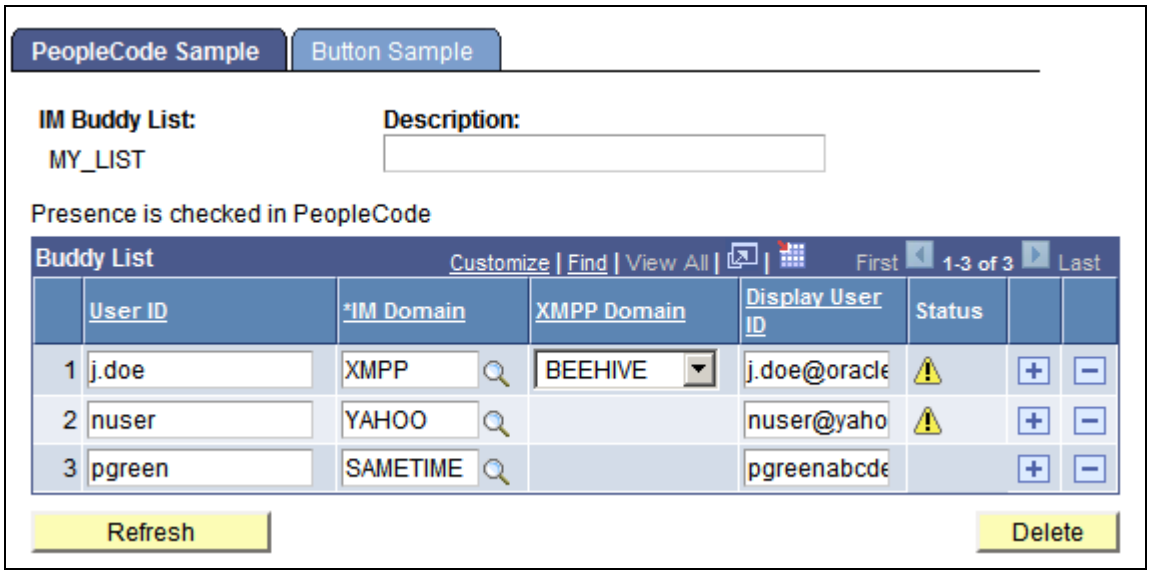
**Note.** XMPP and GTALK presence are not supported using PeopleCode built-ins for presence detection. To check presence for these, use the Button Sample page to check presence.

## Using the Button Sample Page

Access the Button Sample page using the following navigation path:

PeopleTools, MultiChannel Framework, Instant Messaging, Sample Pages, Button Sample


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.




Button Sample page

The Button Sample page demonstrates the use of the instant messaging button implemented in PeopleSoft Application Designer. The sample pages are intended for demonstration purposes and should not be used in production.

See *PeopleTools 8.51 PeopleBook: PeopleSoft Application Designer Developer's Guide*, "Using Page Controls," Specifying Type Properties for Push Buttons or Links.

**IM Buddy List** Displays an identifier for this list.

**Description** Enter a description for this list.

**User and Network** Displays user ID and instant messaging service ID.

**Status** Displays the presence status of the specified user.

Click the icon to initiate an instant messaging session with the selected user. You can send an instant message even if the user is not online.

**Note.** For the SAMETIME icon to be visible, do not select the Sun JVM in the browser. If Sun JVM is selected, the icon does not display in the sample page.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Appendix A

# JSMCAPI Quick Reference

This appendix discusses the constructors, fields, methods, callback event methods, and returns of the JSMCAPI classes.

## JSMCAPI Classes

This section lists the JSMCAPI classes in alphabetical order, along with the constructor, fields, methods, callback event methods, and returns associated with each class.

### _Address

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

#### *Constructor*

| ***Constructor*** | ***Returns*** |
|---|---|
| _Address() | _Address object |

#### *Fields*

| ***Field*** | ***Type*** |
|---|---|
| caps | Object |
| id | String |

#### *Callback Event Methods*

| ***Callback Event Method*** | ***Returns*** |
|---|---|
| onError*(event)* | None |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# _UQAddress

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

The _UQAddress class extends the _Address class.

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

### Constructor

| Constructor | Returns |
|---|---|
| _UQAddress() | _UQAddress object |

### Fields

| Field | Type |
|---|---|
| tasks | Associative array |

### Methods

| Method | Returns |
|---|---|
| acceptTask*(task,reason)* | None |
| dequeueTask*(task)* | Object |

### Callback Event Methods

The _UQAddress class inherits the onError callback event method from the _Address class.

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

| Callback Event Method | Returns |
|---|---|
| onAccepted*(event)* | None |
| onAcceptingTask*(event)* | None |
| onDequeueingTask*(event)* | None |
| onNotify*(event)* | None |
| onTaskAdded*(event)* | None |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Callback Event Method* | *Returns* |
|---|---|
| onTaskRemoved*(event)* | None |
| onUnassigned*(event)* | None |

## _User

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| _User() | _User object |

### *Fields*

| *Field* | *Type* |
|---|---|
| agentId | String |
| caps | Associative array |
| id | String |
| name | String |
| presences | Associative array |
| states | Associative array of the constants beginning with ST_*. |
| ST_LOGGEDIN | String |
| ST_LOGGEDOUT | String |
| ST_NOTREADY | String |
| ST_READY | String |
| ST_UNKNOWN | String |
| ST_WORKNOTREADY | String |
| ST_WORKREADY | String |
| statistics | Object |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Type |
|---|---|
| statistics1 | Object |
| statistics2 | Object |

## A2AChat

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

The A2AChat class extends the _Address class.

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

### Constructor

| Constructor | Returns |
|---|---|
| A2AChat*(event,address,chatType)* | A2AChat object |

### Fields

| Field | Type |
|---|---|
| address | Object |
| agentId | String |
| agentName | String |
| appData | Object |
| chatType | String |
| customerName | Object |
| id | String |
| isConference | Object |
| jr | String of the PS_JR constant. |
| type | String of the following constants:<br>• TYPE_ANSWER<br>• TYPE_CONSULT |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Type |
|---|---|
| uniqueueId | String |

### Methods

| Method | Returns |
|---|---|
| getUrl*(defaultUrl)* | String |

## A2AChatAddress

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| A2AChatAddress() | A2AChatAddress object |

### Fields

The A2AChatAddress class inherits the following fields from the _Address class:

- id
- caps

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

| Field | Type |
|---|---|
| id | String |
| tasks | Associative array |

### Methods

| Method | Returns |
|---|---|
| initiateChat*(agentId)* | None |

### Callback Event Methods

The A2AChatAddress class inherits the onError callback event method from the _Address class.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

| *Callback Event Method* | *Returns* |
|---|---|
| onChatEnded*(event)* | None |
| onInitiatingChat*(event)* | None |
| onNotify*(event)* | None |

## AgentStatistics

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| AgentStatistics() | AgentStatistics object |

### *Fields*

| *Field* | *Type* |
|---|---|
| averageCallDuration | Number |
| averageHoldDuration | Number |
| callsHandled | Number |
| data | Associative array |
| percentIdleTime | Number |
| percentTimeAvailable | Number |
| percentTimeInCurrentState | Number |
| percentTimeUnavailable | Number |
| taskAcceptedCurrentLogin | Number |
| timeCurrentLogin | Number |
| timeWorking | Number |
| totalTaskAcceptedLogin | Number |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Type |
|---|---|
| totalTaskDoneLogin | Number |
| totalTaskDoneLogin | Number |
| unavailableDuration | Number |
| waitDuration | Number |

## AppData

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| AppData() | AppData object |

### Fields

| Field | Type |
|---|---|
| data | Associative array |
| groupId | String |
| jr | String |
| question | String |
| strData | String |
| subject | String |
| uniqueId | String |
| url | String of the constant URL |
| userId | String |
| username | String |
| wizUrl | String |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Methods*

| *Method* | *Returns* |
|---|---|
| addKeyValue*(key,value)* | None |

## Buddy

This section lists the constructor, methods, and callback event methods associated with this class.

The Buddy class extends the _User class.

See Appendix A, "JSMCAPI Quick Reference," _User, page 535.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| Buddy() | Buddy object |

### *Fields*

The Buddy class inherits the following fields from the _User class:

- agentID
- caps
- id
- name
- presences
- states
- statistics
- statistics1
- statistics2

See Appendix A, "JSMCAPI Quick Reference," _User, page 535.

### *Callback Event Methods*

| *Callback Event Method* | *Returns* |
|---|---|
| onStat1*(event)* | None |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Callback Event Method* | *Returns* |
|---|---|
| onStat2*(event)* | None |
| onState*(event)* | None |

## Call

This section lists the constructor, fields, and returns (if applicable) associated with this class.

The Call class extends the Task class.

See Appendix A, "JSMCAPI Quick Reference," Task, page 580.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| Call*(strCall)* | Call object |

### *Fields*

The Call class inherits the following fields from the Task class:

- caseid
- cost
- customerid
- group
- id
- onStat
- priority
- type
- urlAbs
- urlRel

See Appendix A, "JSMCAPI Quick Reference," Task, page 580.

| *Field* | *Type* |
|---|---|
| line | Object |
| statistics | Object |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Type |
|---|---|
| userData | String |

## CallData

This section lists the constructor, fields, methods, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| CallData() | CallData object |

### Fields

| Field | Type |
|---|---|
| ani | String |
| callId | String |
| callType | String |
| data | String |
| dnis | String |

### Methods

| Method | Returns |
|---|---|
| addKeyValue*(key, value)* | None |

## CallStatistics

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| CallStatistics() | CallStatistics object |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Fields

| Field | Type |
|---|---|
| data | Associative array |
| holdTime | String |
| queueTime | String |
| talkTime | String |

## Chat

This section lists the constructor, fields, methods, and returns (if applicable) associated with this class.

The Chat class extends the Task class.

See Appendix A, "JSMCAPI Quick Reference," Task, page 580.

### Constructor

| Constructor | Returns |
|---|---|
| Chat*(event,address)* | Chat object |

### Fields

The Chat class inherits the following fields from the Task class:

- caseid
- cost
- customerid
- group
- id
- onStat
- priority
- type
- urlAbs
- urlRel

See Appendix A, "JSMCAPI Quick Reference," Task, page 580.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Field* | *Type* |
|---|---|
| address | Object |
| agentId | String |
| appData | Object |
| chatconnection | Object |
| chatType | String |
| customerName | String |
| groupId | String |
| question | String |
| statistics | Object |
| subject | String |
| userData | Object |

### *Methods*

| *Method* | *Returns* |
|---|---|
| gettpUrl*(defaultUrl)* | String |
| getUrl*(defaultUrl)* | String |

## ChatAddress

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

The ChatAddress class extends the _UQAddress class.

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| ChatAddress() | ChatAddress object |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Fields

The ChatAddress class inherits the following fields from the _Address class:

- id
- caps

| Field | Returns |
|---|---|
| chatconnections | Object |
| tasks | Object |

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

The ChatAddress class inherits the tasks field from the _UQAddress class.

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

### Methods

The ChatAddress class inherits the following methods from the _UQAddress class:

- acceptTask
- dequeueTask

| Method | Returns |
|---|---|
| chat *(agentId,buddyId,reason)* | None |
| getChatconnectionByConnectionId *(connectionId)* | Object |
| getChatconnectionindexByConnectionId *(connectionId)* | Index |
| getFreeChatconnection | Object |
| getFreeChatconnectionIndex | Object |

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

### Callback Event Methods

The ChatAddress class inherits the onError callback event method from the _Address class.

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

The ChatAddress class inherits the following callback event methods from the _UQAddress class:

- onAccepted

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- onAcceptingTask
- onDequeueingTask
- onNotify
- onTaskAdded
- onTaskRemoved
- onUnassigned

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

| *Callback Event Method* | *Returns* |
|---|---|
| onCapabilitiesChanged *(event)* | None |

## ChatConnection

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| ChatConnection*(event)* | ChatConnection object |

### *Fields*

| *Field* | *Returns* |
|---|---|
| caps | Object |
| chat | String |
| connectionId | String |
| id | String |
| state | String |

### *Methods*

| *Method* | *Returns* |
|---|---|
| answer *(agentid,reason)* | None |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| ***Method*** | ***Returns*** |
|---|---|
| conference *(agentid,reason)* | None |
| forward *(fromagentid,toagentid,qid,userdata,chatdata, reason)* | None |
| gethistory *(agentid,noofLines,reason)* | None |
| getUrl *(deafultURL)* | String |
| message *(agentid,message,reason)* | None |
| pushURL*(URL)* | None |
| reject*(agentid,reason)* | None |
| release*(agentid,reason)* | None |
| typing *(agentid,reason)* | None |

### *Callback Event Methods*

| ***Callback Event Method*** | ***Returns*** |
|---|---|
| onAccepted *(event)* | None |
| onAnswering *(event)* | None |
| onCapabilitiesChanged *(event)* | None |
| onChatdataChanged *(event)* | None |
| onConferencing *(event)* | None |
| onDialing *(event)* | None |
| onDropped *(event)* | None |
| onError *(event)* | None |
| onForwarded *(event)* | None |
| onForwardError *(event)* | None |
| onForwarding *(event)* | None |
| onHistory *(event)* | None |
| onIncomingChat *(event)* | None |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Callback Event Method* | *Returns* |
|---|---|
| onMessage *(event)* | None |
| onPartyAdded *(event)* | None |
| onPartyChanged *(event)* | None |
| onPartyRemoved *(event)* | None |
| onProperties *(event)* | None |
| onPushURL *(event)* | None |
| onRejected *(event)* | None |
| onReleased *(event)* | None |
| onReleasing *(event)* | None |
| onRevoked *(event)* | None |
| onTalking *(event)* | None |
| onTyping *(event)* | None |
| onUserdataChanged *(event)* | None |

## ChatConnectionCaps

This section lists the constructor, fields, methods, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| ChatConnectionCaps*(event)* | ChatConnectionCaps object |

### *Fields*

| *Field* | *Returns* |
|---|---|
| canAnswer | Boolean |
| canConference | Boolean |
| canConferenceSingle | Boolean |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Returns |
|---|---|
| canForward | Boolean |
| canIndicateTyping | Boolean |
| canPushURL | Boolean |
| canReject | Boolean |
| canSendMessage | Boolean |

## ChatData

This section lists the constructor, fields, methods, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| ChatData*(event)* | ChatData object |

### Fields

| Field | Returns |
|---|---|
| data | Object |

### Methods

| Method | Returns |
|---|---|
| addKeyValue*(key,value)* | None |

## Connection

This section lists the constructor, methods, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| Connection *(event)* | ConnectionEvent object |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Methods*

| *Method* | *Returns* |
|---|---|
| eventUserDataChanged*(event)* | None |

## ConnectionListener

This section lists the constructor, methods, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| ConnectionListener *(event)* | ConnectionListener object |

### *Methods*

| *Method* | *Returns* |
|---|---|
| requestAttachUserdata*(object,request)* | None |

## ConnectionRequest

This section lists the constructor, methods, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| ConnectionRequest *(event)* | ConnectionRequest object |

### *Methods*

| *Method* | *Returns* |
|---|---|
| getUserData( ) | UserData |

## Email

This section lists the constructor, fields, methods, and returns (if applicable) associated with this class.

The Email class extends the Task class.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

See Appendix A, "JSMCAPI Quick Reference," Task, page 580.

### Constructor

| Constructor | Returns |
|---|---|
| Email*(event)* | Email object |

### Fields

The Email class inherits the following fields from the Task class:

- caseid
- cost
- customerid
- group
- id
- onStat
- priority
- type
- urlAbs
- urlRel

See Appendix A, "JSMCAPI Quick Reference," Task, page 580.

| Field | Type |
|---|---|
| address | Object |
| agentId | Object |
| appData | Object |
| customerName | String |
| emailconnection | Object |
| emailId | String |
| groupId | String |
| question | String |
| statistics | Object |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Type |
|---|---|
| subject | String |
| userData | Object |

### Methods

| Method | Returns |
|---|---|
| gettpUrl*(defaultUrl)* | Object |
| getUrl*(defaultUrl)* | String |

## EmailAddress

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

The EmailAddress class extends the _UQAddress class.

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

### Constructor

| Constructor | Returns |
|---|---|
| EmailAddress() | EmailAddress object |

### Fields

The EmailAddress class inherits the following fields from the _Address class:

- id
- caps

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

The EmailAddress class inherits the tasks field from the _UQAddress class.

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

| Field | Returns |
|---|---|
| agent | Object |
| emailconnections | Object |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Methods*

The EmailAddress class inherits the following methods from the _UQAddress class:

- acceptTask
- dequeueTask

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

| *Method* | *Returns* |
|---|---|
| getEmailconnectionByConnectionId *(connectionId)* | Object |
| getEmailconnectionindexByConnectionId *(connectionId)* | Index |
| getFreeEmailconnection | Object |

### *Callback Event Methods*

The EmailAddress class inherits the onError callback event method from the _Address class.

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

The EmailAddress class inherits the following callback event methods from the _UQAddress class:

- onAccepted
- onAcceptingTask
- onDequeueingTask
- onNotify
- onTaskAdded
- onTaskRemoved
- onUnassigned

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

## EmailConnection

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| EmailConnection*(event)* | EmailConnection object |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Fields

| Field | Returns |
|---|---|
| caps | Object |
| connectionId | String |
| email | String |
| id | String |
| state | String |

### Methods

| Method | Returns |
|---|---|
| abandon *(reason)* | None |
| answer *(reason)* | None |
| complete *(reason)* | None |
| forward *(fromagentid,toagentid,qid,userdata,Emaildata, reason)* | None |
| reject*(agentid,reason)* | None |
| withdraw_RES *(reason)* | None |

### Callback Event Methods

| Callback Event Method | Returns |
|---|---|
| onAnswering *(event)* | None |
| onCapabilitiesChanged *(event)* | None |
| onCompleted *(event)* | None |
| onDropped *(event)* | None |
| onEmaildataChanged *(event)* | None |
| onError *(event)* | None |
| onForwarded *(event)* | None |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Callback Event Method* | *Returns* |
|---|---|
| onForwardError *(event)* | None |
| onForwarding *(event)* | None |
| onIncoming *(event)* | None |
| onProcessing *(event)* | None |
| onRejected *(event)* | None |
| onRevoked *(event)* | None |
| onUserdataChanged *(event)* | None |
| onWithdraw_REQ *(event)* | None |

## EmailConnectionCaps

This section lists the constructor, fields, methods, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| EmailConnectionCaps*(event)* | EmailConnectionCaps object |

### *Fields*

| *Field* | *Returns* |
|---|---|
| canAnswer | Boolean |
| canComplete | Boolean |
| canForward | Boolean |
| canReject | Boolean |

## EmailData

This section lists the constructor, fields, methods, and returns (if applicable) associated with this class.

### *Constructor*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Constructor | Returns |
|---|---|
| EmailData*(event)* | EmailData object |

### Fields

| Fields | Returns |
|---|---|
| data | Object |

### Methods

| Method | Returns |
|---|---|
| addKeyValue*(key,value)* | None |

## Extension

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

The Extension class extends the _Address class.

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

### Constructor

| Constructor | Returns |
|---|---|
| Extension*(numOfLines)* | Extension object |

### Fields

The Extension class inherits the following fields from the _Address class:

- id
- caps

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

| Field | Type |
|---|---|
| forwardMode | Object |
| isDnd | Boolean |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Type |
|---|---|
| lines | Associative array |
| numOfLines | Number |

### Methods

| Method | Returns |
|---|---|
| cancelDnd*(reason)* | None |
| cancelForwardSet*(reason)* | None |
| forwardSet*(number,mode,reason)* | None |
| getDialingLine() | Object |
| getFreeLine() | Object |
| getLineByConnectionId*(connectionId)* | Object |
| getOffHookLine() | Object |
| setDnd*(reason)* | None |

### Callback Event Methods

The Extension class inherits the onError callback event method from the _Address class.

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

| Callback Event Method | Returns |
|---|---|
| onCancelingDnd*(event)* | None |
| onCancelingForward*(event)* | None |
| onDnd*(event)* | None |
| onDndCanceled*(event)* | None |
| onForwardCanceled*(event)* | None |
| onForwarded*(event)* | None |
| onForwarding*(event)* | None |
| onSettingDnd*(event)* | None |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## ExtensionCaps

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| ExtensionCaps*(strCaps)* | ExtensionCaps object |

### *Fields*

| *Field* | *Type* |
|---|---|
| canCancelDnd | Boolean |
| canDial | Boolean |
| canFwdBusy | Boolean |
| canFwdBusyNoAnswer | Boolean |
| canFwdCancelForward | Boolean |
| canFwdDefault | Boolean |
| canFwdNoAnswer | Boolean |
| canFwdUnconditional | Boolean |
| canRefreshState | Boolean |
| canSetDnd | Boolean |

## ForwardMode

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| ForwardMode() | ForwardMode object |

### *Fields*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Type |
|---|---|
| mode | String of the following constants:<br>• BUSY<br>• BUSYNOANSWER<br>• DEFAULT<br>• NOANSWER<br>• NONE<br>• UNCONDITIONAL |

## GenericAddress

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

The GenericAddress class extends the _UQAddress class.

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

### Constructor

| Constructor | Returns |
|---|---|
| GenericAddress() | GenericAddress object |

### Fields

The GenericAddress class inherits the following fields from the _Address class:

- id
- caps

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

The GenericAddress class inherits the tasks field from the _UQAddress class.

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

| Field | Returns |
|---|---|
| agent | Object |
| genericconnections | Object |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Methods*

The GenericAddress class inherits the following methods from the _UQAddress class:

- acceptTask
- dequeueTask

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

| *Method* | *Returns* |
|---|---|
| getGenericconnectionByConnectionId *(connectionId)* | Object |
| getGenericconnectionindexByConnectionId *(connectionId)* | Index |
| getFreeGenericconnection | Object |

### *Callback Event Methods*

The GenericAddress class inherits the onError callback event method from the _Address class.

See Appendix A, "JSMCAPI Quick Reference," _Address, page 533.

The GenericAddress class inherits the following callback event methods from the _UQAddress class:

- onAccepted
- onAcceptingTask
- onDequeueingTask
- onNotify
- onTaskAdded
- onTaskRemoved
- onUnassigned

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 534.

## GenericConnection

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

### *Constructor*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| ***Constructor*** | ***Returns*** |
|---|---|
| GenericConnection*(event)* | GenericConnection object |

### *Fields*

| ***Field*** | ***Returns*** |
|---|---|
| caps | Object |
| connectionId | String |
| generic | String |
| id | String |
| state | String |

### *Methods*

| ***Method*** | ***Returns*** |
|---|---|
| abandon *(reason)* | None |
| answer *(reason)* | None |
| complete *(reason)* | None |
| forward *(fromagentid,toagentid,qid,userdata,Genericdata, reason)* | None |
| reject*(agentid,reason)* | None |
| withdraw_RES *(reason)* | None |

### *Callback Event Methods*

| ***Callback Event Method*** | ***Rreturns*** |
|---|---|
| onCapabilitiesChanged *(event)* | None |
| onCompleted *(event)* | None |
| onDropped *(event)* | None |
| onError *(event)* | None |
| onForwarded *(event)* | None |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Callback Event Method* | *Rreturns* |
|---|---|
| onForwardError *(event)* | None |
| onForwarding *(event)* | None |
| onGenericdataChanged *(event)* | None |
| onIncoming *(event)* | None |
| onProcessing *(event)* | None |
| onRejected *(event)* | None |
| onRevoked *(event)* | None |
| onUserdataChanged *(event)* | None |
| onWithdraw_REQ *(event)* | |

## GenericConnectionCaps

This section lists the constructor, fields, methods, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| GenericConnectionCaps*(event)* | GenericConnectionCaps object |

### *Fields*

| *Field* | *Returns* |
|---|---|
| canAnswer | Boolean |
| canComplete | Boolean |
| canForward | Boolean |
| canReject | Boolean |

## GenericData

This section lists the constructor, fields, methods, and returns (if applicable) associated with this class.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Constructor

| Constructor | Returns |
|---|---|
| GenericData*(event)* | GenericData object |

### Fields

| Field | Returns |
|---|---|
| data | Object |

### Methods

| Method | Returns |
|---|---|
| addKeyValue*(key,value)* | None |

## GenericTask

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

The GenericTask class extends the Task class.

See Appendix A, "JSMCAPI Quick Reference," Task, page 580.

### Constructor

| Constructor | Returns |
|---|---|
| GenericTask*(event)* | GenericTask object |

### Fields

The GenericTask class inherits the following fields from the Task class:

- caseid
- cost
- customerid
- group
- id
- onStat

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

- priority
- type
- urlAbs
- urlRel

See Appendix A, "JSMCAPI Quick Reference," Task, page 580.

| *Field* | *Type* |
|---|---|
| address | String |
| agentId | String |
| appData | Object |
| customerName | String |
| genericconnection | Object |
| genericId | String |
| groupId | String |
| question | String |
| subject | String |
| userData | Object |

### *Methods*

| *Method* | *Returns* |
|---|---|
| gettpUrl*(defaultUrl)* | Object |
| getUrl*(defaultUrl)* | String |

## GLOBALS

This section lists the fields and methods associated with this class.

### *Fields*

| *Field* | *Field Type* |
|---|---|
| A2AChat.PS_JR | Constant |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Field Type |
|---|---|
| A2AChat.TYPE_ANSWER | Constant |
| A2AChat.TYPE_CONSULT | Constant |
| Server.TYPE.CTI | Constant |
| Server.TYPE_UQ | Constant |
| Task.TYPE_A2ACHAT | Constant |
| Task.TYPE_CHAT | Constant |
| Task.TYPE_CTI | Constant |
| Task.TYPE_EMAIL | Constant |
| Task.TYPE_GENERIC | Constant |

### Methods

| Method | Returns |
|---|---|
| initJSMCAPI() | None |
| isValid*(obj)* | Boolean |
| MCFBroadcast*(cluster,cluster,cluster,cluster,cluster, cluster,cluster,cluster,cluster,cluster)* | None |

## Group

This section lists the constructor, fields, callback event methods, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| Group() | Group object |

### Fields

| Field | Type |
|---|---|
| id | String |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Type |
|---|---|
| name | String |
| registered | Boolean |
| statistics | Object |
| statistics1 | Object |
| statistics2 | Object |

### Callback Event Methods

| Callback Event Methods | Returns |
|---|---|
| onStat*(event)* | None |
| onStat1*(event)* | None |
| onStat2*(event)* | None |
| onTaskAdded*(event)* | None |
| onTaskRemoved*(event)* | None |

## GroupStatistics

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| GroupStatistics() | GroupStatistics object |

### Fields

| Field | Type |
|---|---|
| data | Associative array |
| listOfTasksInTheQueueByTaskType | Number |
| maxTaskCompletionTime | Number |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Field* | *Type* |
|---|---|
| newestTask | Number |
| newestTaskCompletionTime | Number |
| numberOfAbandoned | String |
| numberOfLoggedIn | String |
| numberOfQueued | String |
| numUnassignedTasks | Number |
| queuedWaitTime | String |
| queueUpTime | Number |
| relativeQueueLoad | String |
| timeElapsedOldestTask | Number |

## GroupStatistics1

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| GroupStatistics1*(data)* | GroupStatistics1 object |

### *Fields*

| *Field* | *Type* |
|---|---|
| mostRecentTaskDone | String |
| mostRecentTaskDoneData | String |
| mostRecentTaskEnqueued | String |
| mostRecentTaskEnqueuedData | String |
| numAgentsAvailable | String |
| numAgentsLoggedIn | String |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Type |
|---|---|
| numEscalation | String |
| numOverflow | String |
| numTaskAccepted | String |
| numTaskDone | String |
| numTaskQueued | String |
| reasonFlag | String |
| taskTotalTimeInSystem | String |
| timeSinceStart | String |

## GroupStatistics2

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| GroupStatistics2*(data)* | GroupStatistics2 object |

### Fields

| Field | Type |
|---|---|
| averageTaskDuration | String |
| averageWaitTime | String |
| oldestTask | String |
| recentTask | String |
| timeElapsedOldestTask | String |
| timeElapsedRecentTask | String |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Line

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

### *Constructor*

| ***Constructor*** | ***Returns*** |
|---|---|
| Line() | Line object |

### *Fields*

| ***Field*** | ***Type*** |
|---|---|
| call | Object |
| caps | Object |
| connectionId | String |
| id | String |
| isMuted | Boolean |
| state | String of the following constants:<br>• ST_DIALING<br>• ST_DROPPED<br>• ST_HELD<br>• ST_IDLE<br>• ST_OFFHOOK<br>• ST_RINGING<br>• ST_TALKING |

### *Methods*

| ***Method*** | ***Returns*** |
|---|---|
| alternate*(reason)* | None |
| answer*(reason)* | None |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| ***Method*** | ***Returns*** |
|---|---|
| attachUserData*(userdata,reason)* | None |
| clear*(reason)* | None |
| complete*(reason)* | None |
| conference*(destination,reason,userdata,calldata)* | None |
| conferenceSingle*(destination,reason,userdata,calldata)* | None |
| dial*(number,reason,userdata)* | None |
| dropParty*(destination,reason)* | None |
| getAni() | String |
| getDescr() | String |
| getDnis() | String |
| getReferenceId() | String |
| getUrl*(defaultUrl)* | String |
| grabCall*(destination,reason)* | String |
| hold*(reason)* | String |
| join *(reason,conferenceId )* | None |
| mute *(reason)* | None |
| park*(destination,reason)* | None |
| reconnect*(reason)* | None |
| reject*(reason)* | None |
| release*(reason)* | None |
| retrieve*(reason)* | None |
| sendDTMF*(reason,* stringDTMF) | None |
| setcallresult*(result)* | None |
| setcallresultDNC*(reason,number)* | None |
| setcallresultReschedule*(hours,minutes,day,month,year)* | None |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| ***Method*** | ***Returns*** |
|---|---|
| transfer*(destination,reason,userdata,calldata)* | None |
| transferMute*(destination,reason,userdata,calldata)* | None |
| unmute*(reason)* | None |
| updateCallData*(calldata,reason)* | None |

### *Callback Event Methods*

| ***Callback Event Method*** | ***Returns*** |
|---|---|
| onAlternating*(event)* | None |
| onAnswering*(event)* | None |
| onAttachingUD*(event)* | None |
| onCallDataChanged*(event)* | None |
| onCapabilitiesChanged*(event)* | None |
| onClearing*(event)* | None |
| onCompleting*(event)* | None |
| onConferencing*(event)* | None |
| onDialing*(event)* | None |
| onDropped*(event)* | None |
| onError*(event)* | None |
| onGrabbing*(event)* | None |
| onHeld*(event)* | None |
| onHolding*(event)* | None |
| onJoining*(event)* | None |
| onMuted*(event)* | None |
| onOffHook*(event)* | None |
| onOnHook*(event)* | None |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Callback Event Method* | *Returns* |
|---|---|
| onParking*(event)* | None |
| onPartyAdded*(event)* | None |
| onPartyChanged*(event)* | None |
| onPartyRemoved*(event)* | None |
| onReconnecting*(event)* | None |
| onRejected*(event)* | None |
| onRejecting*(event)* | None |
| onReleasing*(event)* | None |
| onRetrieving*(event)* | None |
| onRinging*(event)* | None |
| onSetcallresult*(event)* | None |
| onSetcallresultDNC*(event)* | None |
| onSetcallresultReschedule*(event)* | None |
| onTalking*(event)* | None |
| onTransfering*(event)* | None |
| onUnmuted*(event)* | None |
| onUpdatingCD*(event)* | None |
| onUserDataChanged*(event)* | None |

## LineCaps

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| LineCaps*(strCaps)* | LineCaps object |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### *Fields*

| *Field* | *Type* |
|---|---|
| canAlternate | Boolean |
| canAnswer | Boolean |
| canAttachUserData | Boolean |
| canClear | Boolean |
| canComplete | Boolean |
| canConference | Boolean |
| canConferenceSingle | Boolean |
| canDropParty | Boolean |
| canHold | Boolean |
| canMute | Boolean |
| canPark | Boolean |
| canReconnect | Boolean |
| canReject | Boolean |
| canRelease | Boolean |
| canRetrieve | Boolean |
| canSendDTMF | Boolean |
| canSetcallresult | Boolean |
| canSetcallresultDNC | Boolean |
| canSetcallresultReschedule | Boolean |
| canTransfer | Boolean |
| canTransferMute | Boolean |
| canUnmute | Boolean |
| canUpdateCallData | Boolean |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## MCEvent

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| MCEvent() | MCEvent object |

### *Fields*

| *Field* | *Type* |
|---|---|
| extension | Object |
| group | Object |
| reason | Object |
| user | Object |

## MediaType

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| MediaType() | MediaType object |

### *Fields*

| *Field* | *Type* |
|---|---|
| type | String of the following constants:<br>• MT_EMAIL<br>• MT_VOICE |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## PSMC

This section lists the constructor, fields, methods, and returns (if applicable) associated with this class.

### *Constructor*

| ***Constructor*** | ***Returns*** |
|---|---|
| PSMC() | PSMC object |

### *Fields*

| ***Field*** | ***Type*** |
|---|---|
| renServer | Object |
| servers | Associative array |
| sessions | Associative array |

### *Methods*

| ***Method*** | ***Returns*** |
|---|---|
| closeSession*(serverId)* | None |
| getCallById*(id)* | Object |
| getChatById*(id)* | Object |
| getEmailById*(id)* | Object |
| getGenericTaskById*(id)* | Object |
| getLineById*(id)* | Object |
| openSession*(serverId)* | None |
| start() | None |
| stop() | None |

## Reason

This section lists the constructor, fields, and returns (if applicable) associated with this class.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Constructor

| Constructor | Returns |
|---|---|
| Reason*(code,desc)* | Reason object |

### Fields

| Field | Type |
|---|---|
| code | String |
| description | String |
| reasonData1 | String |
| reasonData2 | String |
| reasonData3 | String |

## RenServer

This section lists the constructor, fields, callback event methods, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| RenServer() | RenServer object |

### Fields

| Field | Type |
|---|---|
| isRunning | Boolean |
| url | String |

### Callback Event Methods

| Callback Event Method | Returns |
|---|---|
| onDown*(event)* | None |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Callback Event Method | Returns |
|---|---|
| onUp*(event)* | None |

## Server

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| Server*(serverID)* | Server object |

### Fields

| Field | Type |
|---|---|
| id | String |
| info | String |
| state | String of the following constants:<br>• ST_INSERVICE<br>• ST_OUTOFSERVICE |
| type | String of the following constants:<br>• TYPE_CTI<br>• TYPE_UQ |

### Callback Event Methods

| Callback Event Method | Returns |
|---|---|
| onBroadcast*(event)* | None |
| onHbLost*(event)* | None |
| onHbRecovered*(event)* | None |
| onInService*(event)* | None |
| onOutOfService*(event)* | None |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Callback Event Method* | *Returns* |
|---|---|
| onRestart*(event)* | None |

## Session

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| Session*(serverID)* | Session object |

### *Fields*

| *Field* | *Type* |
|---|---|
| addresses | Associative array |
| buddies | Associative array |
| groups | Associative array |
| id | String |
| intervalBetweenReqs | Number |
| numberRegsPerBulkReq | Number |
| serverID | String |
| state | String of the following constants:<br>• ST_ACTIVE<br>• ST_CLOSED<br>• ST_CLOSING<br>• ST_IDLE |
| user | User object |

### *Methods*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| ***Method*** | ***Returns*** |
|---|---|
| broadcastSubscribe*(cluster,queue,task,state,presence, method)* | None |
| broadcastUnsubscribe*(type)* | None |
| close() | None |
| open() | None |
| registerAddress*(address)* | None |
| registerBuddy*(buddy)* | None |
| registerBuddiesBulk*(buddies,numRegsPerBulkReq, intervalBetweenReqs)* | None |
| registerGroup*(group)* | None |
| registerGroupsBulk*(groups,numRegsPerBulkReq, intervalBetweenReqs)* | None |
| registerUser*(user)* | None |
| setAutoRecovery*(autoRecover)* | None |
| statPublish*(userStat,userStat)* | None |
| unregisterAddress*(address)* | None |
| unregisterBuddy*(buddy)* | None |
| unregisterGroup*(group)* | None |
| unregisterUser*(user)* | None |

### *Callback Event Methods*

| ***Callback Event Method*** | ***Returns*** |
|---|---|
| onAddressRegistered*(event)* | None |
| onAddressUnregistered*(event)* | None |
| onBuddyRegistered*(event)* | None |
| onBuddyUnregistered*(event)* | None |
| onClosed*(event)* | None |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Callback Event Method* | *Returns* |
|---|---|
| onError*(event)* | None |
| onGroupRegistered*(event)* | None |
| onGroupUnregistered*(event)* | None |
| onInfo*(event)* | None |
| onOpened*(event)* | None |
| onUserRegistered*(event)* | None |
| onUserUnregistered*(event)* | None |

## Task

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| Task() | Task object |

### *Fields*

| *Field* | *Type* |
|---|---|
| caseid | String |
| cost | String |
| customerid | String |
| group | String |
| id | String |
| onStat | String |
| priority | String |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Field | Type |
|---|---|
| type | String; one of the following:<br>• TYPE_A2ACHAT<br>• TYPE_CHAT<br>• TYPE_CTI<br>• TYPE_EMAIL<br>• TYPE_GENERIC |
| urlAbs | Object |
| urlRel | Object |

### Methods

| Method | Returns |
|---|---|
| setUserData*(UserDatadata)* | None |
| getUserData( ) | UserData |

## TaskStatistics

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| TaskStatistics() | TaskStatistics object |

### Fields

| Field | Type |
|---|---|
| data | Associative array |
| holdTime | String |
| queueTime | String |
| talkTime | String |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# User

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

The User class extends the _User class.

See Appendix A, "JSMCAPI Quick Reference," _User, page 535.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| User*(id,name,agentId,agentPassword)* | User object |

### *Fields*

The User class inherits the following fields from the _User class:

- agentId
- caps
- id
- name
- presences
- states
- statistics
- statistics1
- statistics2

See Appendix A, "JSMCAPI Quick Reference," _User, page 535.

| *Field* | *Type* |
|---|---|
| addresses | Associative array |
| agentPassword | String |
| language | String |
| registeredAddresses | Associative array |
| statesUq | Object |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

### Methods

| Method | Returns |
|---|---|
| disableMedia*(group,mediaType,extension,reason)* | None |
| enableMedia*(group,mediaType,extension,reason)* | None |
| isMediaEnabled*(mediaTypegroup)* | None |
| login*(group, extension,reason)* | None |
| loginUq*(group)* | None |
| logout*(group, extension,reason)* | None |
| logoutUq*(group)* | None |
| register*(extension,reason)* | None |
| setNotReady*(group,presence,extension,reason)* | None |
| setPresence*(group,presence,extension,reason)* | None |
| setPresenceUq*(group,presenceText,reason)* | None |
| setReady*(group,presence,extension,reason)* | None |
| setWorkNotReady*(group,presence,extension,reason)* | None |
| setWorkReady*(group,presence,extension,reason)* | None |
| unRegister*(extension,reason)* | None |

### Callback Event Methods

| Callback Event Method | Returns |
|---|---|
| onCapabilitiesChanged*(event)* | None |
| onCtiBusy*(event)* | None |
| onCtiClear*(event)* | None |
| onDropped*(event)* | None |
| onError*(event)* | None |
| onInfo*(event)* | None |

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| ***Callback Event Method*** | ***Returns*** |
|---|---|
| onLoggedIn*(event)* | None |
| onLoggedOut*(event)* | None |
| onLoggingIn*(event)* | None |
| onLoggingOut*(event)* | None |
| onMediaDisabled*(event)* | None |
| onMediaEnabled*(event)* | None |
| onNotReady*(event)* | None |
| onPresenceChanged*(event)* | None |
| onReady*(event)* | None |
| onRegistered*(event)* | None |
| onRegistering*(event)* | None |
| onSettingNotReady*(event)* | None |
| onSettingPresence*(event)* | None |
| onSettingReady*(event)* | None |
| onSettingWorkNotReady*(event)* | None |
| onSettingWorkReady*(event)* | None |
| onStat*(event)* | None |
| onStat1*(event)* | None |
| onStat2*(event)* | None |
| onUnkown*(event)* | None |
| onUnregistered*(event)* | None |
| onUnregistering*(event)* | None |
| onWorkNotReady*(event)* | None |
| onWorkReady*(event)* | None |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## UserCaps

This section lists the constructor, fields, method, and returns (if applicable) associated with this class.

### *Constructor*

| *Constructor* | *Returns* |
|---|---|
| UserCaps() | UserCaps object |

### *Fields*

| *Field* | *Returns* |
|---|---|
| canHandleChat | Boolean |
| canHandleEmail | Boolean |
| canHandleGeneric | Boolean |
| canHandleVoice | Boolean |
| canLogin | Boolean |
| canLogout | Boolean |
| canRefreshState | Boolean |
| canSetNotReady | Boolean |
| canSetPresence | Boolean |
| canSetReady | Boolean |
| canSetWorkNotReady | Boolean |
| canSetWorkReady | Boolean |

## UserData

This section lists the constructor, constants, methods, and returns (if applicable) associated with this class.

### *Constructor*

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| *Constructor* | *Returns* |
|---|---|
| UserData() | UserData object |

### *Constants*

The UserData class uses the following constants:

- COMPONENT
- DESCR
- ISCRIPTPROGRAMNAME
- ICTYPE
- MARKET
- MENU
- REFERENCEID
- TARGET
- URLABS
- URLREL

### *Methods*

| *Method* | *Returns* |
|---|---|
| add*(key, value)* | None |
| addKeyValue*(key,value)* | None |
| getValue(String key) | String |
| remove*(key)* | None |
| getKeys( ) | String array |

## UserStatistics1

This section lists the constructor, fields, and returns (if applicable) associated with this class.

### *Constructor*


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| Constructor | Returns |
|---|---|
| UserStatistics1() | UserStatistics1 object |

### Fields

| Field | Type |
|---|---|
| availableCost | String |
| ctiBusy | String |
| currentQueue | String |
| mostRecentTaskData | String |
| mostRecentTaskId | String |
| numTaskAccepted | String |
| numTasksDone | String |
| numTasksUnassigned | String |
| presenceText | String |
| reasonFlag | String |
| state | String |
| timeInCurrentState | String |
| timeSinceLoggedIn | String |

## UserStatistics2

This section lists the constructor, fields, methods, callback event methods, and returns (if applicable) associated with this class.

### Constructor

| Constructor | Returns |
|---|---|
| UserStatistics2() | UserStatistics2 object |

### Fields

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

| ***Field*** | ***Type*** |
|---|---|
| currentQueue | String |
| timeIdle | String |
| timeInCurrentState | String |
| timeNotReady | String |
| timeSinceLogin | String |
| totalTimeAvailable | String |
| totalTimeUnavailable | String |


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

## Appendix B

# Installing Digital Certificates for REN SSL

Digital certificates are required to provide client and server authentication for REN SSL. A digital certificate is an electronic means of establishing your credentials for web or business transactions that are issued by a certification authority (CA). The CA is a trusted third party who signs and issues the certificates for users after verifying their authentication using secure means.

This appendix presents a sample way of installing digital certificates and configure REN SSL. PeopleSoft customers may have their own means of obtaining and installing digital certificates for REN SSL.

## Installing Digital Certificates

This section outlines the basic steps to install digital certificates. Before installing digital certificates, you must create the application server domain.

---

**Note.** The application server domain must have write permissions. All certificates are stored under <PS_HOME>/appserv/<domain name>. The cacerts file has write permissions under <PS_HOME>/JRE/lib/security.

---

The following overview lists the steps that are required to install digital certificates. The subsequent sections describe each step in detail.

To install digital certificates and configure REN SSL:

1. Install the CA server certificate.

   See Appendix B, "Installing Digital Certificates for REN SSL," Installing the CA Server Certificate, page 590.

2. Install the REN server certificate.

   See Appendix B, "Installing Digital Certificates for REN SSL," Installing the REN Server Certificate, page 590.

3. Configure digital certificates.

   See Appendix B, "Installing Digital Certificates for REN SSL," Configuring Digital Certificates, page 591.

4. Import certificates into Java keystore.

   See Appendix B, "Installing Digital Certificates for REN SSL," Importing Certificates in Java Keystore, page 592.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

5. Configure the REN server.

   See Appendix B, "Installing Digital Certificates for REN SSL," Configuring the REN Server, page 592.

6. Configure REN clusters.

   See Appendix B, "Installing Digital Certificates for REN SSL," Configuring REN Clusters, page 593.

7. Install certificates for local node.

   See Appendix B, "Installing Digital Certificates for REN SSL," Installing Certificates for Local Node, page 593.

8. Generate the client certificate.

   See Appendix B, "Installing Digital Certificates for REN SSL," Generating the Client Certificate, page 594.

9. Install PSMCAPI certificates.

   See Appendix B, "Installing Digital Certificates for REN SSL," Installing PSMCAPI Certificates, page 595.

## Installing the CA Server Certificate

To install the CA server certificate:

1. Generate the RSA private key for certificate authority.
2. Generate the Certificate Signing Request (CSR) for certificate authority.
3. Generate the PEM file.

   **Note.** If a CA certificate already exists in PEM format, the preceding three steps can be omitted.

4. Import the CA Certificate in PEM format using PeopleTools, Security, Security Objects, Digital Certificates.

The preceding steps are explained in detail in the section Configuring Digital Certificates.

See Appendix B, "Installing Digital Certificates for REN SSL," Configuring Digital Certificates, page 591 and *PeopleTools 8.51 PeopleBook: PeopleSoft Integration Broker Administration*, "Setting Up Secure Integration Environments," Implementing Client Authentication.

## Installing the REN Server Certificate

To install the REN server certificate:

1. Generate REN server CSR using PeopleTools, Security, Security Objects, Digital Certificates.
2. Get the CSR signed by a CA.

   **Note.** The certificate must be in PEM format.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

3. Import the certificate into PeopleTools, Security, Security Objects, Digital Certificates.

The preceding steps are explained in detail in the section Configuring Digital Certificates.

See Appendix B, "Installing Digital Certificates for REN SSL," Configuring Digital Certificates, page 591.

## Configuring Digital Certificates

Before configuring digital certificates, you must generate the private keys, CSR, and PEM file.

To configure digital certificates:

1. Select PeopleTools, Security, Security Objects, Digital Certificates.
2. Click +.
3. Select *ROOTCA* from the Type drop-down list box.
4. Enter an alias name for the CA in Alias, and click Add Root.

   The Add Root Certificate dialog box appears.
5. Open the ca.pem file.

   The root CA certificate is generated.
6. Copy the contents of the ca.pem file, paste them into the Add Root Certificate dialog box, and click OK.
7. Click +.
8. Select *Cert* from the Type drop-down list box.
9. Enter an alias name in Alias, such as PSFTCA.
10. Click Add Root.
11. Select the CA certificate alias of step 4 from the Issuer Alias lookup button.
12. Click Request.

    The Request New Certificate dialog box appears.
13. Complete the Common Name,Org Unit (organization unit),Organization,Locality,State/Province,Country, Algorithm,Key Size,Email Address, and Challenge Pswdfields.

    **Note.** The common name must be the machine name of the REN server machine, for example, PTA112.peoplesoft.com, where PTA112 is the machine name and .peoplesoft.com is the domain name.
14. Click OK.

    The Certificates Signing Request dialog box appears.
15. Copy and paste the text from the Certificates Signing Request dialog box, and save the text in a file named ren.csr in <PS_HOME>\appserv\<domain name>\.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

16. Click OK.

    The Import link appears.

17. Submit ren.csr to the CA that issued the selected root certificate.

    The CA may send you the signed public key certificate by email or require you to download it from a specified web page.

18. Open the saved certificate file in a text editor, and then highlight and copy its entire contents.

19. Select PeopleTools, Security, Security Objects, Digital Certificates.

20. Click Import.

    The Import Certificate page appears.

21. Paste the copied certificate content into the long edit box, and click OK.

See *PeopleTools 8.51 PeopleBook: PeopleSoft Integration Broker Administration*, "Setting Up Secure Integration Environments," Implementing Node Authentication and *PeopleTools 8.51 PeopleBook: PeopleSoft Integration Broker Administration*, "Setting Up Secure Integration Environments," Implementing Nonrepudiation.

## Importing Certificates in Java Keystore

To import certificates in Java keystore:

1. Open a command prompt.
2. Enter the following command:

```
<PS_HOME>\jre\bin\keytool -import -trustcacerts -alias <alias-name>
-file <CA Certificate Pem file> -keystore <PS_HOME>\jre\lib\security\cacerts
-storepass changeit
```

   Example:

```
<PS_HOME>\jre\bin\keytool -import -trustcacerts -alias PSFTCA -file ca.pem
-keystore <PS_HOME>\jre\lib\security\cacerts -storepass changeit
```

---

**Note.** You will get an error message, sslv3 alert certificate unknown, if the certificate is not imported correctly.

---

See *PeopleTools 8.51 PeopleBook: PeopleSoft Integration Broker Administration*, "Setting Up Secure Integration Environments," Understanding Client Authentication.

## Configuring the REN Server

To configure the REN server for SSL:

1. Select PeopleTools, REN Server Configuration, REN Server Definition.

   The REN Server Definition page appears.

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

2. Select the SSL Only check box.
3. Select *RENSERVER* from the Certificate Alias drop-down list box.
4. Click *Save*.

#### *See Also*

Chapter 5, "Configuring REN Servers," Defining REN Servers, page 77

## Configuring REN Clusters

To configure REN clusters:

1. Select PeopleTools, REN Server Configuration, REN Server Cluster.

   The REN Server Cluster page appears.
2. Update the REN server cluster URL using https and the SSL port.
3. Update the REN server browser URL using https and the SSL port.
4. Click Save.

#### *See Also*

Chapter 5, "Configuring REN Servers," Defining a REN Server Cluster, page 80

## Installing Certificates for Local Node

Apart from the CA and REN server certificates, client authentication requires local node certificates, a client certificate for the browser, and a PSMCAPI certificate.

To install certificates for the local node:

1. Select PeopleTools, Security, Security Objects, Digital Certificates.
2. Click +.
3. Select *Local Node* from the Type drop-down list box.
4. Enter the local node name in Alias, and click Add Root.
5. Select the alias of the CA certificate from the Issuer Alias lookup button.
6. Click Request.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved. 

7. Complete the Common Name,Org Unit,Organization,Locality,State/Province,Country,Algorithm,Key Size,Email Address, and Challenge Pswd fields.

   **Note.** The common name must be the machine name of the REN server machine, for example, PTA112.peoplesoft.com, where PTA112 is the machine name and .peoplesoft.com is the domain name.

8. Click OK.

   The Certificates Signing Request dialog box appears.

9. Copy and paste the text from the Certificates Signing Request dialog box, and save the text in a file named qelocal.csr in <PS_HOME>\appserv\<domain name>\.

   **Note.** The file name qelocal.csr is used as an example only.

10. Click OK.

    The Import link appears.

11. To obtain your local node certificate, submit the qelocal.csr to the CA that issued the selected root certificate.

    The process of obtaining digital certificates varies, depending on the CA. Typically, a CA requires you to paste the content of the PEM-formatted CSR into a form that you submit online. The CA may send you the signed public key certificate by email or require you to download it from a specified web page.

12. Open the saved certificate file in a text editor, and then highlight and copy its entire contents.

13. Select PeopleTools, Security, Security Objects, Digital Certificates.

14. Click the Import link.

    The Import Certificate page appears.

15. Paste the copied certificate content into the long edit box, and click OK.

See *PeopleTools 8.51 PeopleBook: PeopleSoft Integration Broker Administration*, "Setting Up Secure Integration Environments," Implementing Node Authentication and *PeopleTools 8.51 PeopleBook: PeopleSoft Integration Broker Administration*, "Setting Up Secure Integration Environments," Implementing Nonrepudiation.

## Generating the Client Certificate

You can generate the client certificate by openssl or keytool in P12 format and import it in the browser. Importing the certificates depends on the browser.

The following steps are an example of generating a client certificate using openssl. Clients can use keytool or Microsoft CA to generate the client certificate.

To generate the client certificate using openssl:

 Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

1. Generate the RSA private key.

   ```
   openssl genrsa -des3 -out <Private key file>
   ```

   Example:

   ```
   openssl genrsa des3 out renclient.key passout pass:pass 1024
   ```

2. Generate a CSR file.

   ```
   openssl req -config <filename> -new -M;key <Private key file> -out <CSR file>
   ```

   Example:

   ```
   openssl req -config ..\apps\openssl.cnf -new -key renclient.key -x509⇒
    -days 365 -out renclient.csr
   ```

3. Generate a PEM format file.

   ```
   openssl x509 -req -days 365 in <CSR file> -CA <CA PEM File> -CAkey<CA
   Key File> -M;CAcreateserial -out <RenServer PEM file> -outform PEM
   ```

   ---

   **Note.** <CA key file> and <CA PEM file> are Certificate Authority Key file and Certificate Authority in PEM format respectively.

   ---

   Example:

   ```
   openssl x509 req days 365 in RENCLIENT.csr -CA ca.pem -CAkey ca.key -⇒
   CAcreateserial -out renclient1.pem -outform PEM -passout pass:pass
   ```

4. Generate a .p12 certificate for the browser.

   ```
   openssl pkcs12 -export -in <RenServer PEM file>  -out <.p12 file> -inkey
   <Private key file> -name <alias name>
   ```

   Example:

   ```
   openssl pkcs12 -export -in renclient.pem -out renclient.p12 -inkey
   renclient.key -M;name renclient
   ```

## Installing PSMCAPI Certificates

To install PSMCAPI certificates:

1. Generate a private key in the keystore using the following command:

   ```
   <PS_HOME>\jre\bin\keytool genkey dname CN=Company Name,
   OU=Organization Unit, O=M;Organization, L=Locality, S=State/Provenance,
   C=Country alias <alias Name>  -M;keyalg RSA validity 365 keystore
   <PS_HOME>\jre\lib\cacerts storepass changeit keypass password
   ```

2. Generate the CSR using the following command:

   ```
   PS_HOME>\jre\bin\keytool certreq alias <alias name> -file <certificate
   file name> -keystore <PS_HOME>\jre\lib\security\cacerts storepass
   changeit keypass password
   ```

3. To obtain your certificate, submit the CSR to the CA that issued the selected root certificate.

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

4. Import the signed certificates into Java keystore using the following command:

```
<PS_HOME>\jre\bin\keytool import alias <alias name> -file <certificate
file .pem> -keystore <PS_HOME>\jre\lib\security\cacerts storepass changeit
keypass password
```

---

**Note.** The clients must import the CA certificate in Java keystore of JRE of PSMCAPI for SSL communication with the REN server using the KeyTool command.

---

See *PeopleTools 8.51 PeopleBook: PeopleSoft Integration Broker Administration*, "Setting Up Secure Integration Environments," Implementing Client Authentication.


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

# Index

## Numerics/Symbols


_Address() constructor 533
_Address callback event methods 533
_Address class
    callback event method 266
    constructor 264
    fields 265
_Address class fields
    caps 265
    id 265
_Address constructor 265, 533
_Address fields 533
_Address object 533
_UQAddress() constructor 534
_UQAddress class
    callback event methods 534
    constructor 266, 534
    event method
        *See Also* _UQAddress class callback event methods
    fields 267, 534
    methods *See Also* _UQAddress methods, 534
_UQAddress class callback event methods
    onAccepted 268
    onAcceptingTask 269
    onDequeueingTask 269
    onNotify 269
    onTaskAdded 269
    onTaskRemoved 270
    onUnassigned 270
_UQAddress class constructor 266
_UQAddress class fields 267
_UQAddress methods
    acceptTask 267
    dequeueTask 268
_UQAddress object 534
_User() constructor 535
_User class 272
    constructor 271, 535
    fields 271, 535
_User class field
    ST_LOGGEDIN 273
    states 272
_User class fields
    agentID 271
    caps 272
    id 272
    name 272
    presences 272
    ST_LOGGEDOUT 273
    ST_NOTREADY 273
    ST_READY 273
    ST_UNKNOWN 273
    ST_WORKNOTREADY 273
    ST_WORKREADY 274
    statistics 274
    statistics1 274
    statistics2 274
_User constructor 271
_User object 535


## A


A2AChat.PS_JR 381, 564
A2AChat.TYPE_ANSWER 382, 565
A2AChat.TYPE_CONSULT 382, 565
A2AChatAddress class
    callback event methods 537
    constructor 279, 537
    event method
        *See Also* A2AChatAddress class callback event methods
    fields 280, 537
    method 281
    methods 537
A2AChatAddress class callback event methods
    onChatEnded 282
    onInitiatingChat 282
    onNotify 282
A2AChatAddress class fields
    id 280
    tasks 281
A2AChatAddress constructor 280, 537
A2AChatAddress object 537
A2AChat class
    constructor 275, 536
    fields 275, 536
    method 279
    methods 537
A2AChat class fields
    address 276
    agentID 276
    agentName 276
    appData 276
    chatType 277
    customerName 277
    id 277
    isConference 278
    jr 278
    question 278
    subject 278
    type 278
    uniqueueId 279
A2AChat class method 279
A2AChat constructor 275, 536
A2AChat object 536
abandon
    EmailConnection class methods 337
    GenericConnection class methods 366
acceptTask 267, 534
ACD *See* Automatic Call Distributor (ACD)
addKeyValue
    AppData class 290, 540
    CallDataclass 542
    CallData class 296
    ChatData class 549
    ChatData class methods 327
    EmailData class 556
    EmailData class methods 347
    GenericData class 563
    GenericData class methods 376
    UserData class 494


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

address
A2AChat class 276, 536
Chat class 299, 544
Email class 328, 551
GenericTask class 378, 564
addresses
Session class 450, 578
User class 470, 582
agent
EmailAddress class fields 333
Email class 552
GenericAddress class 559
GenericAddress class fields 361
agent-specific URL 226
agent chat window 153
agent console *See* MultiChannel Console
Agent Console page 192
agentId
_User class 535
A2AChat class 536
Chat class 300, 544
Email class 329, 551
GenericTask class 378, 564
agentID
_User class 271
A2AChat class 276
Agent Info page 43
agentName 276, 536
Agent page 120
agentPassword 471, 582
agents
activating/viewing chats 152
assigning tasks *See Also* tasks
assigning to queues 120
communicating by chat 7
communicating via chat 153
configuring CTI agents 38
configuring MCF agents 119
configuring windows 125
creating 120
customizing presence 122
defining 119
defining agent-specific URLs 128
defining for third party 220
defining optional characteristics 123
defining presence 227
deleting 120
deleting queues 121
displaying number of agents logged into queues 110
inviting to chats 155, 234
logging the status 111
monitoring 195
moving 134
moving physical queues 135
personalizing chat 127
selecting buddies 152
selecting state masks 34
setting maximum timeouts for queue server logout 115
setting maximum workload 121
setting presence 152
setting skill level 121
setting trace level 130
setting up buddy lists 124
specifying languages 229
specifying log/chat/buddy list names 120
specifying miscellaneous parameters 129
specifying supported languages 121
specifying URL for third party 226
viewing the state summary 139
Agent State Summary component (MCF_AGENTSTATE_CMP) 137
Agent State Summary page 139
agent statistics
monitoring 110
AgentStatistics() constructor 538
AgentStatistics class
constructor 283, 538
fields *See Also* AgentStatistics class fields, 538
AgentStatistics class fields
averageCallDuration 283
averageHoldDuration 284
callsHandled 284
data 284
percentIdleTime 284
percentTimeAvailable 284
percentTimeInCurrentState 285
percentTimeUnavailable 285
timeCurrentLogin 285
timeWorking 285
totalTaskAcceptedLogin 285
totalTaskDoneLogin 286
totalTaskUnassignedLogin 286
unavailableDuration 286
waitDuration 286
AgentStatistics constructor 283
AgentStatistics object 538
alternate 401, 569
ani 295, 542
answer 401, 569
ChatConnection class 546
ChatConnection class methods 310
EmailConnection class 554
EmailConnection class methods 337
GenericConnection class 561
GenericConnection class methods 366
appData
A2AChat class 276, 536
Chat class 300, 544
Email class 329, 551
GenericTask class 378, 564
AppData class 288
constructor 286, 539
fields *See Also* AppData class fields, 539
methods 289, 540
AppData class fields
data 287
groupId 287
jr 288
question 288
strData 288
subject 288
uniqueId 288
url 289
userId 289
username 289
wizUrl 289
AppData constructor 287, 539
AppData object 539
application servers
chat architecture *See Also* chat
configuring MCF servers *See Also* MCF servers
describing MCF cluster architecture 90
logging 147


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

notifying MCF clusters of shutdown 117
specifying the authentication domain 76
understanding REN servers
*See Also* REN servers
using queue servers *See Also* queue servers
Aspect 31, 57
attachUserData 402, 570
ChatConnection class methods 310, 338, 367
authentication, server and client 68
authentication domain 83
configuring REN servers 74, 76
Automatic Call Distributor (ACD)
configuring CTI queues 37
CTI architecture *See Also* CTI
switching agent ready status 61
viewing the ACD group in the CTI console 56
availableCost 495, 587
Avaya Definity G3 31
averageCallDuration 283, 538
averageHoldDuration 284, 538
averageTaskDuration 397, 568
averageWaitTime 397, 568


# B


backup queue servers 96
Balance Queue page 136
bcastinterval 110
binding, sockets 77
broadcast
configuring 159
configuring using JSMCAPI 161
implementing 161
publishing 161
subscribing 161
viewing logs 141
broadcast logs
viewing for third party 238
Broadcast page 141
broadcastSubscribe 453, 579
broadcastUnsubscribe 453, 579
browsers
authentication domain 74
browsers required for CTI 21
configuring REN server security 69
describing REN server cluster requests 67
MCF browser windows 8
MCF cluster browser URLs 96
MultiChannel Console 8
setting REN server browser URLs 83
using MCF cluster browser URLs 95
buddies 451, 578
Buddy class
callback event methods 540
constructor 290, 540, 541
event method
*See Also* Buddy class callback event methods
fields 540
Buddy class callback event methods
onStat1 291
onStat2 292
onState 292
Buddy constructor 290, 540
Buddy List page 124
buddy lists
selecting buddies in the MultiChannel Console 152
setting up for agents 124
setting up for third party 223
Buddy object 540
buffer tests, REN server 82, 96
built-in functions, PeopleCode 201
Button Sample page 530


# C


call 399, 569
Line class 569
callback event
onNotify
_UQAddress class 534
onTaskRemoved 535
callback event methods
_Address class 266, 533
_UQAddress class 268, 534
A2AChatAddress class 282, 537
Buddy class 291, 540
ChatAddress class 307, 545
ChatConnection class 317, 547
EmailAddress class 335, 553
EmailConnection class 341, 554
Extension class 354, 557
GenericAddress class 560
GenericConnection class 561
Group class 388, 566
Line class 418, 571
onAccepted 534
onAcceptingTask 534
onAddressRegistered 579
onAddressUnregistered 579
onAlternating 571
onAnswering
ChatConnection class 547
EmailConnection class 554
onAttachingUD 571
onBroadcast 577
onBuddyRegistered 579
onBuddyUnregistered 579
onCallDataChanged 571
onCancelingDnd 557
onCancelingForward 557
onCapabilitiesChanged
ChatAddress class 546
ChatConnection class 547
EmailConnection class 554
GenericConnection class 561
onChatdataChanged
ChatConnection class 547
onChatEnded 538
onClearing 571
onClosed 579
onCompleted
EmailConnection class 554
GenericConnection class 561, 562
onCompleting 571
onConferencing
ChatConnection class 547
onCtiBusy 583
onCtiClear 583
onDequeueingTask 534


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

onDialing
ChatConnection class 547
onDnd 557
onDndCanceled 557
onDown 576
onDropped
ChatConnection class 547
EmailConnection class 554
GenericConnection class 561
onError
_Address class 533
ChatConnection class 547
EmailConnection class 554
GenericConnection class 561
User class 583
onForwardCanceled 557
onForwarded
ChatConnection class 547
EmailConnection class 554
GenericConnection class 561
onForwardError
ChatConnection class 547
EmailConnection class 555
GenericConnection class 562
onForwarding
ChatConnection class 547
EmailConnection class 555
GenericConnection class 562
onGrabbing 571
onGroupRegistered 580
onGroupUnregistered 580
onHbLost 577
onHbRecovered 577
onHeld 571
onHistory
ChatConnection class 547
onHolding 571
onIncoming
EmailConnection class 555
GenericConnection class 562
onIncomingChat
ChatConnection class 547
onInfo 580, 583
onInitiatingChat 538
onInService 577
onJoining 571
onLoggedIn 584
onLoggedOut 584
onLoggingIn 584
onLoggingOut 584
onMediaDisabled 584
onMediaEnabled 584
onMessage
ChatConnection class 548
onMuted 571
onNotify
A2AChatAddress class 538
onNotReady 584
onOffHook 571
onOnHook 571
onOpened 580
onOutOfService 577
onParking 572
onPartyAdded
ChatConnection class 548
onPartyChanged
ChatConnection class 548
onPartyRemoved
ChatConnection class 548
onPresenceChanged 584
onProperties
ChatConnection class 548
onPushURL
ChatConnection class 548
onReady 584
onReconnecting 572
onRegistered 584
onRegistering 584
onRejected
ChatConnection class 548
EmailConnection class 555
GenericConnection class 562
onRejecting 572
onReleased
ChatConnection class 548
onReleasing
ChatConnection class 548
onRestart 578
onRetrieving 572
onRevoked
ChatConnection class 548
EmailConnection class 555
GenericConnection class 562
onRinging 572
onSetcallresult 572
onSetcallresultDNC 572
onSettingDnd 557
onSettingNotReady 584
onSettingPresence 584
onSettingReady 584
onSettingWorkNotReady 584
onSettingWorkReady 584
onStat
Group class 566
User class 584
onStat1
Buddy class 540
Group class 566
onStat1(
User class 584
onStat2
Buddy class 541
Group class 566
User class 584
onState 541
onTalking
ChatConnection class 548
onTaskAdded
_UQAddress class 534
onTaskRemoved 566
onTransfering 572
onTyping
ChatConnection class 548
onUnassigned
_UQAddress class 535
onUnknown 584
onUnmuted 572
onUnregistered 584
onUnregistering 584
onUp 577
onUpdatingCD 572
onUserdataChanged
ChatConnection class 548
EmailConnection class 555
GenericConnection class 562
onUserDataChanged 572


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

onUserRegistered 580
onUserUnregistered 580
onWorkNotReady 584
onWorkReady 584
RenServer class 445, 576
Server class 448, 577
Session class 460, 579
User class 481, 583
Call class
constructor 292
fields *See Also* Call class fields, 541
Call class fields 293
line 293
statistics 294
Call constructor 292, 541
CallData class
constructor 294, 542
fields 295, 542
method 296
methods 542
CallData class fields
ani 295
callId 295
callType 295
data 295
dnis 296
CallData constructor 294, 542
CallData object 542
callId 295, 542
Call object 541
calls
answering 58
completing 63
CTI architecture *See Also* CTI
dialing outbound 62
disconnecting callers 61
holding/retrieving 61
initiating conference calls 60
selecting actions in the CTI console 56
selecting agent status for incoming 61
selecting masks 34
testing outbound 48
transferring callers 58
understanding call actions 56
callsHandled 284, 538
CallStatistics class
constructor 296, 542
fields 297, 543
CallStatistics class fields
data 297
holdTime 297
queueTime 298
talkTime 298
CallStatistics constructor 297, 542
CallStatistics object 542
callType 295, 542
canAlternate 428, 573
canAnswer 428, 573
ChatConnectionCaps class 548
ChatConnectionCaps class fields 324
EmailConnectionCaps class 555
EmailConnection class fields 345
GenericConnectionCaps class 562
GenericConnectionCaps class fields 374
canAttachUserData 573
canAttachUserDate 428
canCancelDnd 357, 558
canCancelDndFwdDefault 558
cancelDnd 350, 557
cancelForwardSet 350, 557
canClear 428, 573
canComplete 429, 573
EmailConnection class fields 346
GenericConnectionCaps class fields 374
canConference 429, 573
ChatConnectionCaps class 548
ChatConnectionCaps class fields 324
EmailConnectionCaps class 555
GenericConnectionCaps class 562
canConferenceSingle 429, 573
ChatConnectionCaps class 548
ChatConnectionCaps class fields 325
canDial 357, 558
canDropParty 429, 573
canForward
ChatConnectionCaps class 549
ChatConnectionCaps class fields 325
EmailConnectionCaps class 555
EmailConnection class fields 346
GenericConnectionCaps class 562
GenericConnectionCaps class fields 375
canFwdBusy 357, 558
canFwdBusyNoAnswer 358, 558
canFwdCancelForward 358, 558
canFwdDefault 358
canFwdNoAnswer 358, 558
canFwdUnconditional 358, 558
canHandleChat 585
UserCaps class fields 490
canHandleEmail 585
UserCaps class fields 490
canHandleGeneric 585
UserCaps class fields 490
canHandleVoice 585
UserCaps class fields 491
canHold 429, 573
canIndicateTyping
ChatConnectionCaps class 549
ChatConnectionCaps class fields 325
canLogin 491, 585
canLogout 491, 585
canMute 430, 573
canned messages 218
canned URL 219
canPark 430, 573
canPushURL
ChatConnectionCaps class 549
ChatConnectionCaps class fields 325
canReconnect 430, 573
canRefreshState 359, 491, 558, 585
canReject 430, 573
ChatConnectionCaps class 549
ChatConnectionCaps class fields 325
EmailConnectionCaps class 555
EmailConnection class fields 346
GenericConnectionCaps class 562
GenericConnectionCaps class fields 375
canRelease 430, 573
canRetrieve 431, 573
canSendDTMF 431, 573
canSendMessage
ChatConnectionCaps class 549
ChatConnectionCaps class fields 326
canSetcallresult 431, 573
canSetcallresultDNC 431, 573
canSetcallresultReschedule 431, 573


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

canSetDnd 359, 558
canSetNotReady 491, 585
canSetPresence 492, 585
canSetReady 492, 585
canSetWorkNotReady 492, 585
canSetWorkReady 492, 585
canTransfer 432, 573
canTransferMute 432, 573
canUnmute 432, 573
canUpdateCallData 432, 573
caps
  _Address class 265, 533
  _User class 272, 535
  ChatConnection class 546
  ChatConnection class fields 308
  EmailConnection class 554
  EmailConnection class fields 336
  GenericConnection class 561
  GenericConnection class fields 365
  Line class 399
caseid 465, 580
CA server certificate 590
certificate alias 79
certificates
  importing in REN Java client 592
  installing for local node 593
  installing PSMCAPI certificates 595
channels
  communicating by chat *See Also* chat
  email *See Also* email channels
  generic 7
  instant messaging *See Also* instant messaging
  MultiChannel Console 8
  understanding 5
  voice 7
chat
  activating 152
  architecture 9
  ChatAddress class 305, 545
  ChatConnection class 546
  ChatConnection class fields 309
  collaborating by 7
  communicating with agents 153
  communicating with customers 155, 234
  defining responses 99
  ending 155, 157, 234, 236
  forwarding to other queues 155, 234
  initiating 152
  instant messaging *See Also* instant messaging
  inviting agents 155, 234
  logging 93, 113, 214
  moving agents during ongoing sessions 135
  personalizing 127
  responding 155, 156, 234, 235
  selecting task acceptance mode 127
  sending/grabbing URLs 155, 234
  sending template messages 155, 234
  setting up buddy lists 124
  specifying agent names 120
  specifying greeting interval 106
  using in MultiChannel Console 151
  using the Customer Chat Sample page 170
  viewing history 154, 233
  viewing logs 142
ChatAddress class
  callback event methods 545
  constructor 303, 544
  event method 307
  fields 304, 545
  methods 304, 545
ChatAddress class callback event
  onCapabilitiesChanged 307
ChatAddress class fields
  chatconnections 304
ChatAddress class method
  chat 305
  getChatconnectionByConnectionId 305
  getChatconnectionindexByConnectionId 306
  getFreeChatconnection 306
  getFreeChatconnectionIndex 307
ChatAddress constructor 303, 544
ChatAddress object 544
Chat class
  constructor 298, 543
  fields *See Also* Chat class fields, 543
  methods 302, 544
Chat class fields 299
  address 299
  agentId 300
  appData 300
  chatconnection 300
  chatType 300
  customerName 301
  groupId 301
  question 301
  statistics 301
  subject 301
  userData 302
Chat class methods
  gettpUrl 302
  getUrl 303
chatconnection
  Chat class 300, 544
ChatConnection
  attachUserData 310, 338, 367
ChatConnectionCaps class
  constructor 323, 548
  fields
    *See Also* ChatConnectionCaps class fields, 548
ChatConnectionCaps class fields
  canAnswer 324
  canConference 324
  canConferenceSingle 325
  canForward 325
  canIndicateTyping 325
  canPushURL 325
  canReject 325
  canSendMessage 326
ChatConnectionCaps constructor 548
ChatConnectionCaps object 548
ChatConnection class
  callback event methods 317, 547
  constructor 308, 546
  fields 308, 546
  methods 310, 546
ChatConnection class callback event methods
  onAccepted 317
  onAnswering 317
  onCapabilitiesChanged 317
  onChatdataChanged 318
  onConferencing 318
  onDialing 318
  onDropped 318
  onError 319
  onForwarded 319


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

onForwardError 319
onForwarding 319
onHistory 320
onIncomingChat 320
onMessage 320
onPartyAdded 320
onPartyChanged 321
onPartyRemoved 321
onProperties 321
onPushURL 321
onRejected 322
onReleased 322
onReleasing 322
onRevoked 322
onTalking 323
onTyping 323
onUserdataChanged 323
ChatConnection class fields
caps 308
chat 309
connectionId 309
id 309
state 309
ChatConnection class methods
answer 310
conference 311
forward 311
gethistory 312
getUrl 313
message 313
pushURL 314
reject 314
release 315
typing 315
wrapup 316
ChatConnection constructor 546
ChatConnection object 546
chatconnections
ChatAddress class 545
ChatAddress class fields 304
Chat constructor 298, 543
ChatData class
constructor 326, 549
fields *See Also* ChatData class fields, 549
methods
*See Also* ChatData class methods, 549
ChatData class fields
data 326
ChatData class methods
addKeyValue 327
ChatData constructor 549
Chat Log component (MCF_CHAT_LOG_CMP) 140
Chat object 543
Chat page 142
Chat Responses page 99
chatType 277, 536
Chat class 300
Cisco
components required for CTI 21
configuring the expanded call context variable 36
CTI architecture *See Also* CTI
implementing 33
logging out 43
selecting a CTI vendor 31
setting agent state masks 34
setting CTI client signature 39
setting masks 33
setting the heartbeat 34
setting up for pop-ups 45
transferring callers 59
Cisco switch limitations 52
clear 402, 570
clhbinterval 110
client authentication 68
close 453, 579
closeSession 437, 575
cluster-wide broadcast 159
Cluster Notify component (MCF_AD_NOTIFY_CMP) 116
clusters
defining for third party 210
implementing MCF *See Also* MCF clusters
notifying for third party 215
REN server *See Also* REN servers
tuning for third party 212
viewing cluster IDs 218
Cluster Summary component (MCF_RSERV_CFG_CMP) 107
Cluster Summary page 107
Cluster Tuning component (MCF_SYSTEM_NV_CMP) 108
Cluster Tuning page 108
code 443, 576
complete 403, 570
EmailConnection class 554
EmailConnection class methods 339
GenericConnection class 561
GenericConnection class methods 368
computer telephony integration (CTI) *See* CTI
conference 403, 570
ChatConnection class 547
ChatConnection class methods 311
conferenceSingle 404, 570
configuration ID 211, 218
Configuration page (instant messaging) 523
connectionid 399
connectionId 569
ChatConnection class 546
ChatConnection class fields 309
EmailConnection class 554
EmailConnection class fields 336
GenericConnection class 561
GenericConnection class fields 365
Connectors page 505
consoles
agent *See Also* MultiChannel Console
CTI *See Also* CTI Console
Java 40
MultiChannel
*See Also* MultiChannel Console
constructor
_Address class 264, 533
_UQAddress class 266, 534
_User class 271, 535
A2AChatAddress class 279, 537
A2AChat class 536
A2AChat Class 275
AgentStatistics 283
AgentStatistics class 538
AppData class 286, 539
Buddy class 290, 540
Call class 292, 541
CallData class 294, 542
CallStatisticsclass 296


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

CallStatistics class 542
ChatAddress 303
ChatAddress class 544
Chat class 298, 543
ChatConnection 308
ChatConnectionCaps class 323, 548
ChatConnection class 546
ChatData class 326, 549
EmailAddress class 332
Email class 327, 551, 552
EmailConnectionCaps class 345, 555
EmailConnection class 335, 553
EmailData class 346, 555
ExtensionCaps class 356, 558
Extension class 348, 556
ForwardMode class 359, 558
GenericAddress class 360, 559
GenericConnectionCaps class 373, 562
GenericConnection class 364, 560
GenericData class 375, 563
GenericTask class 376, 563
Group class 386, 565
GroupStatistics1 class 392, 567
GroupStatistics2 class 396, 568
GroupStatistics class 390, 566
LineCaps class 427, 572
Line class 398, 569
MCEvent class 432, 574
MediaType class 434, 574
PSMC class 435, 575
Reason class 442, 576
RenServer class 444, 576
Server class 446, 577
Session class 450, 578
Task class 464, 580
TaskStatistics class 467, 581
UserCaps class
489, *See Also* UserCaps class fields, 585
User class 469, 582
UserData class 492, 585
UserStatistics1 class 494, 586
UserStatistics2 class 498, 587
cost 465, 580
CTI
accessing 53
agents *See Also* CTI agents
answering calls 58
applying applet protection 21
Cisco components, required 21
completing calls 63
components 16
configuring pop-up windows 44
configuring queues 37
configuring using CTI server 22
configuring using third party 22
creating configurations 24
describing Cisco switch limitations 52
dialing outbound calls 62
disconnecting callers 61
enabling 24
event logging 47
Genesys components, required 21
holding calls 61
implementing Cisco *See Also* Cisco
implementing free seating 47
implementing Genesys *See Also* Genesys
initiating conference calls 60
installing 22, 23
JRE, required 21
log 113
miscellaneous 22
monitoring 111
personalizing 41
phone books, shared 22
registering with middleware 55
setting pop-up screens 26
supporting single signon 47
testing calls 48
transferring callers 58
understanding 15, 51
understanding call actions 56
understanding components 15
using 52
using console *See Also* CTI console
using hot keys 63
using outbound call sample page 47
using phone books 40
using phone books, shared 25
using sample page 47
using the interface 52
CTI Agent Configuration component (PT_CTI_AGENT) 36
CTI Agent Configuration page 38
CTI agents
configuring 38
implementing wrap-up mode 61
logging out 43, 56
phone books, shared 25
signing on to PeopleSoft 24
switching agent ready status 61
using phone books 40
viewing information 43
viewing status 56
CTI Applet page 24
ctiBusy 495, 587
CTI Busy 190
ctiBusyUq 472
CTI Cisco page 33
CTI configuration component (PT_CTI_CONFIG) 22
CTI console 16
heartbeat interval 110
synchronizing with teleset 59
using 55
CTI Console
applet 22, 30
browser requirements 21
components 15
floating 43
JSMCAPI 22
non-applet 22, 29
understanding 15
CTI Genesys page 32
CTI Non-Applet page 24
CTI queues, configuring 37
CTI Sample console
using 184
CTI Sample Pages component (PT_CTI_DEMOOUTB) 178
CTI Sample Pages component (PT_CTI_DEMOUTB) 47
CTI Type page 24
currentQueue 496, 499, 587, 588
Customer Chat page 170
customer chat window 156
customerid 465, 580



Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

customerName 277, 536
Chat class 301, 544
Email class 329, 551
GenericTask class 378, 564
customers
accepting tasks in chat 127
chat architecture *See Also* chat
communicating by chat 7
communicating via chat 155, 234
using CTI *See Also* CTI
using the Customer Chat Sample page 170

# D

data
AgentStatistics class 284, 538
AppData class 287, 539
CallData class 295, 542
CallStatistics class 297, 543
ChatData class 549
ChatData class fields 326
EmailData class 556
EmailData class fields 347
GenericData class 563
GenericData class fields 376
GroupStatistics class 390, 566
TaskStatistics class 581
TaskStatistics class fields 468
debugging
logging for application servers 147
REN servers 75
tracing level for agents 130
troubleshooting CTI 40
viewing event logs 144
default_http_port 76
default_https_port 76
defining
MCF queues 98
Definity G3, Avaya 31
DeQueue
demonstrating 175, 178
understanding 201
dequeueTask 268, 534
desc 443
description 576
dial 405, 570
digital certificates
configuring 591
describing 68
installing for REN SSL 589
digital signatures 21
disableMedia 472, 583
DND 190
DND ( do not disturb) 245
dnis 296, 542
domain
authenticating domain 83
domains
configuring MCF server *See Also* MCF
describing application server domains
*See Also* application servers
describing MCF cluster architecture 90
setting screen pop-up URLs 27
setting the authentication token domain for REN servers
*See Also* authentication domain
done list, size 111
donelistsize 111
download attachments 510
configuring email for large downloads 510
dropParty 405, 570
DTMFInfo 190
DTMF info 246
dumpagents 111
dumpinterval 111

# E

ECC variables 45
email
accessing/deleting from the database 519
authorizing attachments for viewing/deleting 519
channels *See Also* email channels
defining connector and mail server properties 506
EmailConnection class 554
EmailConnection class fields 336
email IDs 520
GetMail - Server page 517
handling headers, size determination and time zone offsets 502
HTML 7
instant messaging *See Also* instant messaging
MailStore - DB page 519
MIME
*See Also* Multipurpose Internet Mail Extensions (MIME)
processing architecture 11
reading/deleting from mail servers 517
setting access, attachment, language and threshold properties 506
storing in/retrieving from database servers 517
understanding 7
unique identifiers (UIDs) 519
universal queue 7
using sample pages 176
using the Email window 177
EmailAddress class
callback event methods 335, 553
constructor 332
fields
*See Also* EmailAddress class fields, 552
methods
*See Also* EmailAddress class methods, 553
EmailAddress class fields
agent 333
emailconnections 333
EmailAddress class methods
getEmailconnectionByConnectionId 333
getEmailconnectionindexByConnectionId 334
getFreeEmailconnection 334
EmailAddress constructor 332, 552
EmailAddress object 552
email channels
configuring GETMAILTARGET 505
configuring Integration Broker 504
demonstrating 516
understanding 501
Email class

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

constructor 327, 551, 552
fields *See Also* Email class fields, 551
methods *See Also* Email class methods, 552
Email class fields 328
address 328
agentId 329
appData 329
customerName 329
emailconnection 329
emailId 329
groupId 330
question 330
statistics 330
subject 330
userData 330
Email class methods
gettpUrl 331
getUrl 331
emailconnection
Email class 329, 551
EmailConnectionCaps class
constructor 345, 555
fields 555
EmailConnectionCaps constructor 555
EmailConnectionCaps object 555
EmailConnection class
callback event methods
*See Also* EmailConnection class callback event methods, 554
constructor 335, 553
fields
*See Also* EmailConnection class fields, EmailConnection class fields, 554
methods
*See Also* EmailConnection class methods, 554
EmailConnection class callback event methods
onAnswering 341
onCapabilitiesChanged 341
onCompleted 341
onDropped 342
onEmaildataChanged 342
onError 342
onForwarded 342
onForwardError 343
onForwarding 343
onIncoming 343
onProcessing 343
onRejected 344
onRevoked 344
onUserdataChanged 344
onWithdraw_REQ 344
EmailConnection class fields
canAnswer 345
canComplete 346
canForward 346
canReject 346
caps 336
connectionId 336
email 336
id 336
state 336
EmailConnection class methods
abandon 337
answer 337
complete 339
forward 339
reject 340
withdraw_RES 340
EmailConnection constructor 553
EmailConnection object 553
emailconnections
EmailAddress class fields 333
Email class 552
Email constructor 327, 551
EmailData class
constructor 346, 555
fields *See Also* EmailData class fields, 556
methods
*See Also* EmailData class methods, 556
EmailData class fields
data 347
EmailData class methods
addKeyValue 347
EmailData constructor 556
emailId
Email class 329, 551
Email object 549, 551, 556
Email page 176
Email Sample Pages component (MCFEM_DEMOERMS_CMP) 516
Email window 177
enableMedia 473, 583
enableMedialogout 475
enableMediasetWorkNotReady 479
EnQueue
accepting tasks in chat 127
demonstrating generic persistent events 173
setting agent languages 121
understanding 201
errors
determining email size 503
logging for application servers 147
REN servers 75
troubleshooting CTI 40
escalation
administering tasks 149
specifying the default timeout for tasks 106
Escalation Admin (administration) page 149
Escalation Administration component (MCF_ESCAL_CMP) 148
event *See* callback event methods
_Address class callback 266
_UQAddress class callback 268
A2AChatAddress class callback 282
Buddy class callback 291
ChatAddress class callback 307
Extension class callback 354
Group class callback 388
Line class callback 418
RenServer class callback 445
Server class callback 448
Session class callback 460
User class callback 481
Event Log component (MCF_EVENTLOGL_CMP) 140
event logs
viewing for third party 236
events
logging 47
MCF log server failover 93
types 145
understanding real-time event notification
*See Also* REN servers
using Generic Event page 173
using Generic Event window 175



Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

viewing event logs 144
Expanded Call Context (ECC) variables 45
extension 433, 574
ExtensionCaps class
constructor 558
fields
*See Also* ExtensionCaps class fields, 558
ExtensionCaps class fields
canCancelDnd 357
canDial 357
canFwdBusy 357
canFwdBusyNoAnswer 358
canFwdCancelForward 358
canFwdDefault 358
canFwdNoAnswer 358
canFwdUnconditional 358
canRefreshState 359
canSetDnd 359
Extensioncaps constructor 558
ExtensionCaps constructor 356
ExtensionCaps object 558
Extension class
callback event methods 557
constructor 348, 556
event method
*See Also* Extension class callback event methods
fields *See Also* Extension class fields, 556
method 349
methods 557
Extension class callback event methods
onCancelingDnd 354
onCancelingForward 354
onDnd 354
onDndCanceled 355
onForwardCanceled 355
onForwarded 355
onForwarding 356
onSettingDnd 356
Extension class fields 348
forwardMode 349
isDnd 349
lines 349
numOfLines 349
Extension class method
cancelDnd 350
cancelForwardSet 350
forwardSet 351
getDialingLine 351
getFreeLine 352
getLineByConnectionId 352
getOffHookLine 353
setDnd 353
Extension constructor 348, 556
Extension object 556


# F


failover
configuring REN servers 65
REN server clusters 67
understanding 89
using MCF log servers 93
using queue servers 92
fields
_Address class
*See Also* _Address class fields, 533
_UQAddress class 267, 534
_User class *See Also* _User class fields, 535
A2AChat.PS_JR 564
A2AChat.TYPE_ANSWER 565
A2AChat.TYPE_CONSULT 565
A2AChatAddress class
*See Also* A2AChatAddress class fields, 537
A2AChat class 536
A2AChat Class
*See Also* A2AChat class fields
address
A2AChat class 536
Chat class 544
Email class 551
GenericTask class 564
addresses
Session class 578
User class 582
agent
Email class 552
GenericAddress class 559
agentId
_User class 535
A2AChat class 536
Chat class 544
Email class 551
GenericTask class 564
agentName 536
agentPassword 582
AgentStatistics class 283, 538
ani 542
appData
A2AChat class 536
Chat class 544
Email class 551
GenericTask class 564
AppData class 287, 539
availableCost 587
averageCallDuration 538
averageHoldDuration 538
averageTaskDuration 568
averageWaitTime 568
buddies 578
Buddy class 540
call 569
Line class 569
Call class 293, 541
CallData class
*See Also* CallData class fields, 542
callId 542
callsHandled 538
CallStatistics class
*See Also* CallStatistics class fields, 543
callType 542
canAlternate 573
canAnswer
ChatConnectionCaps class 548
EmailConnectionCaps class 555
GenericConnectionCaps class 562
canAttachUserData 573
canCancelDnd 558
canClear 573
canComplete 573
canConference
ChatConnectionCaps class 548
EmailConnectionCaps class 555


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

GenericConnectionCaps class 562
canConferenceSingle
ChatConnectionCaps class 548
canDial 558
canDropParty 573
canForward
ChatConnectionCaps class 549
EmailConnectionCaps class 555
GenericConnectionCaps class 562
canFwdBusy 558
canFwdBusyNoAnswer 558
canFwdCancelForward 558
canFwdDefault 558
canFwdNoAnswer 558
canFwdUnconditional 558
canHandleChat 585
canHandleEmail 585
canHandleGeneric 585
canHandleVoice 585
canHold 573
canIndicateTyping
ChatConnectionCaps class 549
canLogin 585
canLogout 585
canMute 573
canPark 573
canPushURL
ChatConnectionCaps class 549
canReconnect 573
canRefreshState 558, 585
canReject
ChatConnectionCaps class 549
EmailConnectionCaps class 555
GenericConnectionCaps class 562
canRelease 573
canRetrieve 573
canSendDTMF 573
canSendMessage
ChatConnectionCaps class 549
canSetcallresult 573
canSetcallresultDNC 573
canSetcallresultReschedule 573
canSetDnd 558
canSetNotReady 585
canSetPresence 585
canSetReady 585
canSetWorkNotReady 585
canSetWorkReady 585
canTransfer 573
canTransferMute 573
canUnmute 573
canUpdateCallData 573
caps
_Address class 533
_User class 535
ChatConnection class 546
EmailConnection class 554
GenericConnection class 561
caseid 580
chat
ChatConnection class 546
ChatAddress 304
ChatAddress class 545
Chat class 299, 543
chatconnection
Chat class 544
ChatConnection 308
ChatConnectionCaps class 324, 548
ChatConnection class 546
chatconnections
ChatAddress class 545
ChatData class 326, 549
chatType 536
code 576
connectionId
ChatConnection class 546
EmailConnection class 554
GenericConnection class 561
cost 580
ctiBusy 587
currentQueue 587, 588
customerid 580
customerName 536
Chat class 544
Email class 551
GenericTask class 564
data
AgentStatistics class 538
AppData class 539
CallData class 542
CallStatistics class 543
ChatData class 549
EmailData class 556
GenericData class 563
GroupStatistics class 566
TaskStatistics class 581
description 576
dnis 542
email
EmailConnection class 554
EmailAddress class 332, 552
Email class 328, 551
emailconnection
Email class 551
EmailConnectionCaps class 555
EmailConnection class 335, 345, 347, 554
emailconnections
Email class 552
EmailData class 556
emailId
Email class 551
extension 574
ExtensionCaps class 357, 558
Extension class 348, 556
forwardMode 556
ForwardMode class 360, 558
generic
GenericConnection class 561
GenericAddress class 377, 559
genericconnection
GenericTask class 564
GenericConnectionCaps class 374, 562
GenericConnection class 364, 561
genericconnections
GenericAddress class 559
GenericData class 375, 563
genericId
GenericTask class 564
GenericTask class 563
gettpUrl
GenericTask class 564
getUrl
GenericTask class 564
GLOBALS class 381, 564
group 574, 580
Group class 387, 565



Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

groupId 539, 544
Email class 551
GenericTask class 564
groups 578
GroupStatistics1 class 393, 567
GroupStatistics2 class 397, 568
GroupStatistics class 390, 566
holdTime 543
TaskStatistics class 581
id
_Address class 533
_User class 535
A2AChat class 536
A2AChatAddress class 537
ChatConnection class 546
EmailConnection class 554
GenericConnection class 561
Group class 565
Line class 569
Server class 577
Session class 578
Task class 580
info
Server class 577
isConference
A2AChat class 536
isDnd 556
isMuted
Line class 569
isRunning 576
jr
A2AChat class 536
AppData class 539
language 582
line 541
LineCaps class 428, 573
Line class 399, 569
lines 557
listOfTasksInTheQueueByTaskType
GroupStatistics class 566
maxTaskCompletionTime 566
MCEvent class 433, 574
MediaType class 435, 574
mode 559
mostRecentTaskData 587
mostRecentTaskDone 567, 568
mostRecentTaskDoneData 567
mostRecentTaskEnqueued 567
mostRecentTaskEnqueuedData 567
mostRecentTaskId 587
name
_User class 535
Group class 566
newestTask 567
newestTaskCompletionTime 567
numAgentsAvailable 567
numAgentsLoggedIn 567
numberOfAbandoned 567
numberOfLoggedIn 567
numberOfQueued 567
numEscalation 568
numOfLines 557
numOverflow 568
numTaskAccepted 568
UserStatistics1 class 587
numTaskDone 568
numTasksDone 587
numTasksUnassigned 587
numUnassignedTasks 567
oldestTask 568
onStat 580
percentIdleTime
AgentStatistics class 538
percentTimeAvailable
AgentStatistics class 538
percentTimeInCurrentState
AgentStatistics class 538
percentTimeUnavailable
AgentStatistics class 538
presences 535
presenceText 587
priority 580
PSMC class 436, 575
question
AppData class 539
Chat class 544
Email class 551
GenericTask class 564
queuedWaitTime 567
queueTime 543
TaskStatistics class 581
queueUpTime 567
reason 574
Reason class 443, 576
reasonData1 576
reasonData2 576
reasonData3 576
reasonFlag
GroupStatistics1 class 568
UserStatistics1 class 587
recentTask 568
registered
Group class 566
registeredAddresses 582
relativeQueueLoad 567
renServer 575
RenServer class 445, 576
Server.TYPE_UQ 565
Server.TYPE.CTI 565
Server class 446, 577
serverID 578
servers 575
Session class 450, 578
sessions 575
ST_LOGGEDIN 535
ST_LOGGEDOUT 535
ST_NOTREADY 535
ST_READY 535
ST_UNKNOWN 535
ST_WORKNOTREADY 535
ST_WORKREADY 535
state
ChatConnection class 546
EmailConnection class 554
GenericConnection class 561
Line class 569
Server class 577
Session class 578
UserStatistics1 class 587
states 535
statesUq 582
statistics
_User class 535
Call class 541
Chat class 544
Email class 551

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Group class 566
statistics1
_User class 536
Group class 566
statistics2
_User class 536
Group class 566
strData 539
subject
AppData class 539
Chat class 544
Email class 552
GenericTask 564
talkTime 543
TaskStatistics class 581
Task.TYPE_A2ACHAT 565
Task.TYPE_CHAT 565
Task.TYPE_CTI 565
Task.TYPE_EMAIL 565
Task.TYPE_GENERIC 565
taskAcceptedCurrentLogin
AgentStatistics class 538
Task class 465, 580
tasks
_UQAddress class 534
A2AChatAddress class 537
ChatAddress class 545
TaskStatistics class 468, 581
taskTotalTimeInSystem 568
timeCurrentLogin
AgentStatistics class 538
timeElapsedOldestTask 567, 568
timeElapsedRecentTask 568
timeIdle 588
timeInCurrentState 587, 588
timeNotReady 588
timeSinceLoggedIn 587
timeSinceLogin 588
timeSinceStart 568
timeWorking
AgentStatistics class 538
totalTaskAcceptedLogin
AgentStatistics class 538
totalTaskDoneLogin
AgentStatistics class 539
totalTimeAvailable 588
totalTimeUnavailable 588
type
A2AChat class 536
MediaType class 574
Server class 577
Task class 581
unavailableDuration 539
uniqueId 539
uniqueueId 537
url 539, 576
urlAbs 581
urlRel 581
user 574
Session class 578
UserCaps class 585
User class 470, 582
userData 542
Chat class 544
Email class 552
GenericTask 564
UserData class
493, *See Also* UserStatistics1 class fields
userId 539
username 539
UserStatistics1 class 587
UserStatistics2 class
*See Also* UserStatistics2 class fields, 587
waitDuration 539
wizUrl 539
fieldschatconnection class methods
GenericAddress class 361
flag, third party 209
floating CTI Console 41, 43
forward 245
ChatConnection class 547
ChatConnection class methods 311
EmailConnection class 554
EmailConnection class methods 339
GenericConnection class 561
GenericConnection class methods 368
Forward 190
Forward function
demonstrating 178
understanding 201
forwardMode 349, 556
ForwardMode class
constructor 359, 558
fields 360, 558
ForwardMode constructor 359, 558
ForwardMode object 558
forwardSet 351, 557
free seating 47


# G


gateways, configuring for email channels 504
Gateways page 504
generic
GenericConnection class 561
GenericConnection class fields 365
GenericAddress class
callback event methods 363, 560
constructor 360, 559
fields
*See Also* GenericAddress class fields, GenericTask class Fields, 559
methods
*See Also* GenericAddress class methods, 560
GenericAddress class fields
agent 361
genericconnections 361
GenericAddress class methods
getFreeGenericconnection 362
getGenericconnectionByConnectionId 362
getGenericconnectionindexByConnectionId 363
GenericAddress constructor 360, 559
GenericAddress object 559
genericconnection
GenericTask class 379, 564
GenericConnectionCaps class
constructor 373, 562
fields
*See Also* GenericConnectionCaps class fields, 562
GenericConnectionCaps class fields
canAnswer 374
canComplete 374



Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

canForward 375
canReject 375
GenericConnectionCaps constructor 562
GenericConnectionCaps object 562
GenericConnection class
callback event methods
*See Also* GenericConnection class callback event methods, 370, 561
constructor 364, 560
fields
*See Also* GenericConnection class fields, 561
methods
*See Also* GenericConnection class methods , 561
GenericConnection class callback event methods
onCapabilitiesChanged 370
onCompleted 370
onDropped 370
onError 371
onForwarded 371
onForwardError 371
onForwarding 371
onGenericdataChanged 372
onIncoming 372
onProcessing 372
onRejected 372
onRevoked 373
onUserdataChanged 373
onWithdraw_REQ 373
GenericConnection class fields
caps 365
connectionId 365
generic 365
id 365
state 365
GenericConnection class methods
abandon 366
answer 366
complete 368
forward 368
reject 369
withdraw_RES 369
GenericConnection constructor 561
GenericConnection object 561
genericconnections
GenericAddress class 559
GenericAddress class fields 361
GenericData class
constructor 375, 563
fields *See Also* GenericData class fields, 563
methods
*See Also* GenericData class methods, 563
GenericData class fields
data 376
GenericData class methods
addKeyValue 376
GenericData constructor 563
Generic Event page 173
Generic Event window 175
genericId
GenericTask class 379, 564
Generic object 563
GenericTask class
constructor 376, 563
fields 563
method *See Also* GenericTask class methods
methods 564
GenericTask class fields 377
address 378
agentId 378
appData 378
customerName 378
genericconnection 379
genericId 379
groupId 379
question 379
statistics 379
subject 380
userdata 380
GenericTask class methods
gettpUrl 380
getUrl 381
GenericTask constructor 377, 563
GenericTask object 563
Genesys
components required for CTI 21
CTI architecture *See Also* CTI
defining wrap-up time 63
implementing 32
logging out 56
selecting a CTI vendor 31
selecting a switch 31
setting up for pop-ups 45
getAni 406, 570
getCallById 438, 575
getChatById 438, 575
getChatconnectionByConnectionId
ChatAddress class 305, 545
getChatconnectionindexByConnectionId
ChatAddress class 306, 545
getDescr 406, 570
getDialingLine 351, 557
getDnis 407, 570
getEmailById 439, 575
getEmailconnectionByConnectionId
EmailAddress class 553
EmailAddress class methods 333
getEmailconnectionindexByConnectionId
EmailAddress class 553
EmailAddress class methods 334
getFreeChatconnection
ChatAddress class 306, 545
getFreeChatconnectionIndex
ChatAddress class 307, 545
getFreeEmailconnection
EmailAddress class 553
EmailAddress class methods 334
getFreeGenericconnection
GenericAddress class 560
GenericAddress class methods 362
getFreeLine 352, 557
getGenericconnectionByConnectionId
GenericAddress class 560
GenericAddress class methods 362
getGenericconnectionindexByConnectionId
GenericAddress class 560
GenericAddress class methods 363
getGenericTaskById 439, 575
gethistory
ChatConnection class 547
ChatConnection class methods 312
getLineByConnectionId 352, 557
getLineById 440, 575
GetMail - Server page 517
GETMAILTARGET, configuring 505


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

getOffHookLine 353, 557
getPadvalue 407
getReferenceId 407, 570
gettpUrl
  Chat class 544
  Email class 552
  Email class methods 331
  GenericTask class 564
  GenericTask class methods 380
getUrl
  A2AChat class 537
  Chat class 302, 303, 544
  ChatConnection class 547
  ChatConnection class methods 313
  Email class 552
  Email class methods 331
  GenericTask class 381, 564
  Line class 408, 570
getURL
  A2AChat class 279
GLOBALS class
  fields *See Also* GLOBALS class fields, 564
  method 384
  methods 565
GLOBALS class fields
  A2AChat.PS_JR 381
  A2AChat.TYPE_ANSWER 382
  A2AChat.TYPE_CONSULT 382
  Server.TYPE_CTI 382
  Server.TYPE_UQ 382
  Task.TYPE_A2ACHAT 383
  Task.TYPE_CHAT 383
  Task.TYPE_CTI 383
  Task.TYPE_EMAIL 383
  Task.TYPE_GENERIC 384
GLOBALS class methods
  initJSMCAPI 384
  isValid 385
  MCFBroadcast 385
grabCall 408, 570
group 574, 580
  MCEvent class 433
  Task class 466
Group class 387
  callback event methods 566
  constructor 386, 565
  event method
    *See Also* Group class callback event methods
  fields *See Also* Group class fields, 565
Group class callback event methods
  onStat 388
  onStat1 389
  onStat2 389
  onTaskAdded 389
  onTaskRemoved 389
Group class fields
  id 387
  name 387
  registered 387
  statistics 387
  statistics1 388
  statistics2 388
Group constructor 386, 565
groupId 539, 544
  AppData class 287
  Chat class 301
  Email class 330, 551
  GenericTask class 379, 564
groups 451, 578
GroupStatistics1 class 568
  constructor 392, 567
  fields
    *See Also* GroupStatistics1 class fields, 567
GroupStatistics1 class fields
  mostRecentTaskDone 393
  mostRecentTaskDoneData 393
  mostRecentTaskEnqueued 394
  mostRecentTaskEnqueuedData 394
  numAgentsAvailable 394
  numAgentsLoggedIn 394
  numEscalation 394
  numOverflow 395
  numTaskAccepted 395
  numTaskDone 395
  numTaskQueued 395
  reasonFlag 395
  taskTotalTimeInSystem 396
  timeSinceStart 396
GroupStatistics1 constructor 393, 567
GroupStatistics1 object 567
GroupStatistics2 class
  constructor 396, 568
  fields
    *See Also* GroupStatistics2 class fields, 568
GroupStatistics2 class fields
  averageTaskDuration 397
  averageWaitTime 397
  oldestTask 397
  recentTask 397
  timeElapsedOldestTask 398
  timeElapsedRecentTask 398
GroupStatistics2 constructor 396, 568
GroupStatistics2 object 568
GroupStatistics class
  constructor 390, 566
  fields
    *See Also* GroupStatistics class fields, 566
GroupStatistics class fields
  data 390
  listOfTasksInTheQueueByTaskType 390
  maxTaskCompletionTime 391
  newestTask 391
  newestTaskCompletionTime 391
  numberOfAbandoned 391
  numberOfLoggedIn 391
  numberOfQueued 391
  numUnassignedTasks 392
  queuedWaitTime 392
  queueUpTime 392
  relativeQueueLoad 392
  timeElapsedOldestTask 392
GroupStatistics constructor 390, 566
GroupStatistics object 566

# H

heartbeat, setting Cisco 34
heartbeat interval, JSMCAPI client 110
Hicom 300, Siemens 31
highwater 112
hold 409, 570
hold status 61
holdTime 297, 543, 581


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

TaskStatistics class fields 468
hot keys
using CTI hot keys 63
using MultiChannel Console 153

# I

id 272, 387
_Address class 265, 533
_User class 535
A2AChatAddress class 280, 537
A2AChat class 277, 536
ChatConnection class 546
ChatConnection class fields 309
EmailConnection class 554
EmailConnection class fields 336
GenericConnection class 561
GenericConnection class fields 365
Group class 565
Line class 400, 569
Server class 447, 577
Session class 451, 578
Task class 466, 580
IMAP4
*See* Internet Message Access Protocol 4 (IMAP4)
implementation
MultiChannel Framework 1
IMServlet
logging 525
info 447
Server class 577
InitChat
demonstrating 170
setting agent languages 121
understanding 201
initiateChat 281, 537
initJSMCAPI 384, 565
instant messaging 521
configuring 522
configuring servers 522
demonstrating 529, 530
integrating 521
Lotus Sametime Connect
*See Also* Lotus Sametime Connect
understanding 7, *See Also* chat
using sample pages 528
Yahoo! Messenger
*See Also* Yahoo! Messenger, 523
Instant Messaging Configuration component (MCF_IM_CFG_CMP) 522
Instant Messaging Sample Pages component (MCF_IM_DEMO_CMP) 528
Integration Broker
configuring for email channels 504
setting the email size threshold 509
Interactive Voice Response (IVR)
CTI architecture *See Also* CTI
Internet Message Access Protocol 4 (IMAP4)
calculating email size 512
setting the connector threshold 509
intervalBetweenReqs 451
io_buffer_size 75
isConference
A2AChat class 278, 536
isDnd 349, 556
isMediaEnabled 474, 583
isMuted
Line class 400, 569
isRunning 445, 576
isValid 385, 565
IVR *See* Interactive Voice Response (IVR)

# J

Java Console 40
Java Runtime Environment (JRE) *See* JRE
JavaScript MultiChannel API (JSMCAPI)
*See* JSMCAPI, JSMCAPI
join 409
jr 288
A2AChat class 278, 536
AppData class 539
JRE 21
JSMCAPI 16, 20
implementing 178
JSMCAPI broadcast
configuring using MCF Supervisor console 161
understanding 160
JSMCAPI Broadcast page 161
JSMCAPI classes
_Address class 533
_UQAddress class 534
_User class 535
A2AChatAddress class 537
A2AChat class 536
AgentStatistics class 538
AppData class 539
Buddy class 540
Call class 541
CallData class 542
CallStatistics class 542
ChatAddress class 544
Chat class 543
ChatConnectionCaps class 548
ChatConnection class 546
ChatData class 549
EmailAddress class 552
Email class 550
EmailConnectionCaps class 555
EmailConnection class 553
EmailData class 555
ExtensionCaps class 558
Extension class 556
ForwardMode class 558
GenericAddress class 559
GenericConnectionCaps class 562
GenericConnection class 560
GenericData class 562
GenericTask class 563
GLOBALS class 564
Group class 565
GroupStatistics1 class 567
GroupStatistics2 class 568
GroupStatistics class 566
LineCaps class 572
Line class 569
MCEvent class 574
MediaType class 574
PSMC class 575
Reason class 575

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

RenServer class 576
Server class 577
Session class 578
Task class 580
TaskStatistics class 581
UserCaps class 585
User class 582
UserData class 585
UserStatistics1 class 586
UserStatistics2 class 587


# L


language 471, 582
languages
setting agent support for 121
setting for email 510
Languages page 121
line 293, 541
LineCaps class
constructor 427, 572
fields *See Also* LineCaps class fields, 573
LineCaps class fields
canAlternate 428
canAnswer 428
canAttachUserDate 428
canClear 428
canComplete 429
canConference 429
canConferenceSingle 429
canDropParty 429
canHold 429
canMute 430
canPark 430
canReconnect 430
canReject 430
canRelease 430
canRetrieve 431
canSendDTMF 431
canSetcallresult 431
canSetcallresultDNC 431
canSetcallresultReschedule 431
canTransfer 432
canTransferMute 432
canUnmute 432
canUpdateCallData 432
LineCaps constructor 427, 572
LineCaps object 572
Line class
callback event methods 571
constructor 398, 569
event method
*See Also* Line class callback event methods
fields *See Also* Line class fields, 569
method 401
methods 569
Line class callback event methods
onAlternating 418
onAnswering 418
onAttachingUD 419
onCallDataChanged 419
onCapabilitiesChanged 419
onClearing 419
onCompleting 420
onConferencing 420
onDialing 420
onDropped 420
onError 421
onGrabbing 421
onHeld 421
onHolding 421
onJoining 422
onMuted 422
onOffHook 422
onOnHook 422
onParking 423
onPartyAdded 423
onPartyChanged 423
onPartyRemoved 423
onReconnecting 424
onRejected 424
onRejecting 424
onReleasing 424
onRetrieving 425
onRinging 425
onSetcallresult 425
onSetcallresultDNC 425
onSetcallresultReschedule 426
onTalking 426
onTransfering 426
onUnmuted 426
onUpdatingCD 427
onUserDataChanged 427
Line class fields
call 399
caps 399
connectionid 399
id 400
isMuted 400
state 400
Line class methods
alternate 401
answer 401
attachUserData 402
clear 402
complete 403
conference 403
conferenceSingle 404
dial 405
dropParty 405
getAni 406
getDescr 406
getDnis 407
getPadvalue 407
getReferenceId 407
getUrl 408
grabCall 408
hold 409
join 409
mute 410
park 411
reconnect 411
reject 412
release 412
retrieve 413
sendDTMF 413
setcallresult 414
setcallresultDNC 414
setcallresultReschedule 415
transfer 416
transferMute 416
unmute 417
updateCallData 417
Line constructor 398, 569



Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Line object 569
lines 349, 557
listOfTasksInTheQueueByTaskType
GroupStatistics class 390, 566
local node 593
log_broadcast 113, 214
log_chat_ses 113, 214
log_cti 113
logDMPQ 112, 213
LogFence settings 147
logging 502
agent status 111
chat 113, 154, 214, 233
CTI 113
CTI events 47
IMServlet 525
MCF log server failover 93
queue server status 111
REN servers 75, 112, 213
specifying agent names for logs 120
statistics 113, 214
tracing level for agents 130
using application servers 147
viewing broadcast logs 141
viewing chat logs 142
viewing event logs 144
logical queues
assigning agents 120
balancing agents/tasks across 136
deleting 98
implementing scalability 93
moving agents 134
moving physical queues within 135
understanding *See Also* queues
viewing physical queue IDs 99
login 474, 583
loginUq 475, 583
logout 583
logoutUq 476, 583
logStat 113, 214
Lotus Sametime Connect
configuring instant messaging servers 522
detecting presence 530
instant messaging 521
setting the server IP address 523
lowwater 114

# M

mail servers
configuring GETMAILTARGET properties 505
reading/deleting email 517
MailStore - DB page 519
masks, Cisco 33
master and slave servers
determining master MCF server failure 114, 215
implementing MCF log server failover 93
implementing queue server failover 92
masterinterval 114, 215
master servers *See* master and slave servers
max_no_reply 115
max_refresh 115
maxTaskCompletionTime 566
GroupStatistics class 391
MCEvent class
constructor 432, 574
fields *See Also* MCEvent class fields, 574
MCEvent class fields
extension 433
group 433
reason 434
user 434
MCEvent constructor 433, 574
MCEvent object 574
MCF *See* MultiChannel Framework (MCF)
configuring broadcast 159
configuring for third party 203
monitoring agent 195
working with sample pages 169
MCF_ACCPT_TASK_CMP component 133
MCF_AD_NOTIFY_CMP component 116
MCF_AGENT_CMP component 119
MCF_AGENTSTATE_CMP component 137
MCF_CHAT_LOG_CMP component 140
MCF_DEMO_CMP component 169, 178
MCF_ESCAL_CMP component 148
MCF_EVENTLOGL_CMP component 140
MCF_IM_CFG_CMP component 522
MCF_IM_DEMO_CMP component 528
MCF_OVERFLOWL_CMP component 148
MCF_Q_CONFIG_CMP component 97
MCF_QSERVSTATE_CMP component 137
MCF_QUEUESTATE_CMP 137
MCF_RSERV_CFG_CMP component 107
MCF_SYSTEM_NV_CMP component 108
MCF_TASKCFG_CMP component 101
MCF_UQCLUSTER component 94
MCF Agent component (MCF_AGENT_CMP) 119
MCF agents *See* agents
creating for third party 222
defining for third party 220
MCFBroadcast 385
MCF cluster
tuning parameters 212
using third party routing 210
MCF Cluster component (MCF_UQCLUSTER) 94
MCF Cluster page 95
MCF clusters
configuring 89, 94, 95
defining 95
deleting 96
describing architecture 90
implementing queue server scalability 93
notifying for third party 215
notifying of changed parameters 116
tuning parameters 108
using queue servers within 96
viewing cluster IDs 99
viewing summary information 107
MCFEM_DEMOERMS_CMP component 516
MCF log servers
configuring MCF clusters 95
implementing failover 93
MCF Queue component (MCF_Q_CONFIG_CMP) 97
MCF queues
configuring and defining 97
defining 98
defining chat responses 99
defining for third party 217

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

defining static push URLs 100
MCF Sample Console 240
MCF Sample Pages component (MCF_DEMO_CMP) 169, 178
MCF servers
architecture 8
configuring 89, 94
determining master failure 114, 215
MCF Supervisor Console 161
MCF task
configuring for third party 231
MCF Task component (MCF_TASKCFG_CMP) 101
MCF Task Configuration page 102
media, agents 228
MediaType class
constructor 434, 574
fields 435, 574
MediaType constructor 434, 574
MediaType object 574
Meridian, Nortel 31
message
ChatConnection class 547
ChatConnection class methods 313
messages
defining 225
defining canned queue 218
messaging, instant *See* instant messaging
method
_Address class callback event 266
alternate 569
answer 569
attachUserData 570
broadcastSubscribe 579
broadcastUnsubscribe 579
clear 570
close 579
closeSession 575
complete 570
conference 570
conferenceSingle 570
dial 570
disableMedia 583
dropParty 570
enableMedia 583
getAni 570
getCallById 575
getChatById 575
getDescr 570
getDnis 570
getEmailById 575
getGenericTaskById 575
getLineById 575
getReferenceId 570
getUrl
Line class 570
grabCall 570
hold 570
isMediaEnabled 583
login 583
loginUq 583
logout 583
logoutUq 583
open 579
openSession 575
park 570
reconnect 570
register 583
registerAddress 579
registerBuddy 579
registerGroup 579
registerUser 579
reject 570
release 570
retrieve 570
sendDTMF 570
setAutoRecovery 579
setcallresult 570
setcallresultDNC 570
setcallresultReschedule 570
setNotReady 583
setPresence 583
setPresenceUq 583
setReady 583
setWorkNotReady 583
setWorkReady 583
start 575
statPublish 579
stop 575
transfer 571
transferMute 571
unmute 571
unRegister 583
UserData class 586
unregisterAddress 579
unregisterBuddy 579
unregisterGroup 579
unregisterUser 579
updateCallData 571
methods
_UQAddress 267
_UQAddress class 534
_UQAddress class callback event 268
A2AChatAddress class 281, 537
A2AChatAddress class callback event 282
A2AChat class 537
A2AChat Class 279
acceptTask 534
addKeyValue
AppData class 540
CallData class 542
ChatData class 549
EmailData class 556
GenericData class 563
answer
ChatConnection class 546
EmailConnection class 554
GenericConnection class 561
AppData class 289, 540
Buddy class callback event 291
CallData class 296, 542
cancelDnd 557
cancelDndforwardSet 557
cancelForwardSet 557
chat
ChatAddress class 545
ChatAddress class 304, 545
ChatAddress class callback event 307
Chat class 302, 544
ChatConnection 310
ChatConnection class 546
ChatData class 326, 549
complete
EmailConnection class 554
GenericConnection class 561
conference



Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

ChatConnection class 547
dequeueTask 534
EmailAddress class 333, 553
Email class 331, 552
EmailConnection class 337, 347, 554
EmailData class 556
Extension class 557
Extension Class
*See Also* Extension class method
Extension class callback event 354
forward
ChatConnection class 547
EmailConnection class 554
GenericConnection class 561
GenericAddress class 362, 560
GenericConnection class 366, 561
GenericData class 376, 563
GenericTask class 380, 564
getChatconnectionByConnectionId
ChatAddress class 545
getChatconnectionindexByConnectionId
ChatAddress class 545
getDialingLine 557
getEmailconnectionByConnectionId
EmailAddress class 553
getEmailconnectionindexByConnectionId
EmailAddress class 553
getFreeChatconnection
ChatAddress class 545
getFreeChatconnectionIndex
ChatAddress class 545
getFreeEmailconnection
EmailAddress class 553
getFreeGenericconnection
GenericAddress class 560
getFreeLine 557
getGenericconnectionByConnectionId
GenericAddress class 560
getGenericconnectionindexByConnectionId
GenericAddress class 560
gethistory
ChatConnection class 547
getLineByConnectionId 557
getOffHookLine 557
gettpUrl
Chat class 544
Email class 552
getUrl
A2AChat class 537
Chat class 544
ChatConnection class 547
Email class 552
GLOBALS class
*See Also* GLOBALS class methods, 565
Group class callback event 388
initiateChat 537
initJSMCAPI 565
isValid 565
Line class *See Also* Line class methods, 569
Line class callback event 418
message
ChatConnection class 547
PSMC class
*See Also* PSMC class methods, 575
pushURL
ChatConnection class 547
reject
ChatConnection class 547
EmailConnection class 554
GenericConnection class 561
release
ChatConnection class 547
RenServer class callback event 445
Server class callback event 448
Session class
*See Also* Session class methods, 578
Session class callback event 460
setDnd 557
typing
ChatConnection class 547
User class *See Also* User class methods, 583
User class callback event 481
UserData class 494, 586
withdraw_RES
EmailConnection class 554
GenericConnection class 561
MIME
*See* Multipurpose Internet Mail Extensions (MIME)
Miscellaneous component (PT_CTI_MISC) 22
Miscellaneous page 26, 129
miscellaneous parameters 230
mode 559
mode field 360
Monitor Agents page 195
Monitor Queues page 197
mostRecentTaskData 496, 587
mostRecentTaskDone 393, 567, 568
mostRecentTaskDoneData 393, 567
mostRecentTaskEnqueued 394, 567
mostRecentTaskEnqueuedData 394, 567
mostRecentTaskId 496, 587
Move Agent page 134
Move Queue page 135
MultiChannel Console
channels 7
CTI interface 52
displaying logged agents/unassigned tasks per queue 110
managing tasks/chat 151
understanding 8
using hot keys 153
MultiChannel Framework (MCF)
administering 133
configuring 94
configuring Broadcast 159
configuring failover 89
configuring scalability 89
elements 5
getting started 1
implementation 1
overview 1
server architecture 8
understanding 5
working with sample pages 159
Multipurpose Internet Mail Extensions (MIME) 7, 512
mute 245, 410
Mute 189

# N

name
_User class 535

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

_User class fields 272
Group class 387, 566
NameValue Pairs
implementing JSMCAPI 164
implementing PeopleCode 167
newestTask 567
GroupStatistics class 391
newestTaskCompletionTime 567
GroupStatistics class 391
nodes
configuring for email channels 504
configuring for email languages 510
configuring MCF_GETMAIL properties 505
Nortel Meridian 31
notifications
notifying clusters of changed parameters 116
understanding real-time event
*See Also* REN servers
Notify Cluster page 116
notifyinterval 215
NotifyQ
demonstrating generic persistent events 173
understanding 202
numAgentsAvailable 394, 567
numAgentsLoggedIn 394, 567
numberOfAbandoned 391, 567
numberOfLoggedIn 391, 567
numberOfQueued 391, 567
numEscalation 394, 568
numOfLines 349, 557
numOverflow 395, 568
numTaskAccepted 568
GroupStatistics1 class 395
UserStatistics1 class 496, 587
numTaskDone 395, 568
numTaskQueued 395
numTasksDone 496, 587
numTasksUnassigned 497, 587
numUnassignedTasks 392, 567


# O


Object 533
oldestTask 397, 568
onAccepted 534
ChatConnection class callback event methods 317
onAccepted event 268
onAcceptingTask 269, 534
onAddressRegistered 461, 579
onAddressUnregistered 461, 579
onAlternating 418, 571
onAnswering 418, 571
ChatConnection class 547
ChatConnection class callback event methods 317
EmailConnection class 554
EmailConnection class callback event methods 341
onAttachingUD 419, 571
onBroadcast 448, 577
onBuddyRegistered 461, 579
onBuddyUnregistered 461, 579
onCallDataChanged 419, 571
onCancelingDnd 354, 557
onCancelingForward 354, 557
onCapabilitiesChanged 419, 481, 571, 583
ChatAddress class 546
ChatAddress class callback event 307
ChatConnection class 547
ChatConnection class callback event methods 317
EmailConnection class 554
EmailConnection class callback event methods 341
GenericConnection class 561
GenericConnection class callback event methods 370
onChatdataChanged
ChatConnection class 547
ChatConnection class callback event methods 318
onChatEnded 282, 538
onClearing 419, 571
onClosed 462, 579
onCompleted
EmailConnection class 554
EmailConnection class callback event methods 341
GenericConnection class 561, 562
GenericConnection class callback event methods 370
onCompleting 420, 571
onConferencing 420, 571
ChatConnection class 547
ChatConnection class callback event methods 318
onCtiBusy 482, 583
onCTIBUSYUq 482
onCtiClear 482, 583
onDequeueingTask 269, 534
onDialing 420, 571
ChatConnection class 547
ChatConnection class callback event methods 318
onDnd 354, 557
onDndCanceled 355, 557
onDown 445, 576
onDropped 571, 583
ChatConnection class 547
ChatConnection class callback event methods 318
EmailConnection class 554
EmailConnection class callback event methods 342
GenericConnection class 561
GenericConnection class callback event methods 370
Line class 420
User class 482
onEmaildataChanged
EmailConnection class callback event methods 342
onError 571, 580
_Address class 266, 533
ChatConnection class 547
ChatConnection class callback event methods 319
EmailConnection class 554
EmailConnection class callback event methods 342
GenericConnection class 561
GenericConnection class callback event methods 371



Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Line class 421
Session class 462
User class 483, 583
onForwardCanceled 355, 557
onForwarded 355, 557
ChatConnection class 547
ChatConnection class callback event methods 319
EmailConnection class 554
EmailConnection class callback event methods 342
GenericConnection class 561
GenericConnection class callback event methods 371
onForwardError
ChatConnection class 547
ChatConnection class callback event methods 319
EmailConnection class 555
EmailConnection class callback event methods 343
GenericConnection class 562
GenericConnection class callback event methods 371
onForwarding 356, 557
ChatConnection class 547
ChatConnection class callback event methods 319
EmailConnection class 555
EmailConnection class callback event methods 343
GenericConnection class 562
GenericConnection class callback event methods 371
onGenericdataChanged
GenericConnection class callback event methods 372
onGrabbing 421, 571
onGroupRegistered 462, 580
onGroupUnregistered 462, 580
onHbLost 448, 577
onHbRecovered 449, 577
onHeld 421, 571
onHistory
ChatConnection class 547
ChatConnection class callback event methods 320
onHolding 421, 571
onIncoming
EmailConnection class 555
EmailConnection class callback event methods 343
GenericConnection class 562
GenericConnection class callback event methods 372
onIncomingChat
ChatConnection class 547
ChatConnection class callback event methods 320
onInfo 580, 583
Session class 463
User class 483
onInitiatingChat 282, 538
onInService 449, 577
onJoining 422, 571
onLoggedIn 483, 584
onLoggedOut 483, 584
onLoggingIn 484, 584
onLoggingOut 484, 584
onMediaDisabled 484, 584
onMediaEnabled 484, 584
onMessage
ChatConnection class 548
ChatConnection class callback event methods 320
onMuted 422, 571
onNotify
_UQAddress class 269, 534
A2AChatAddress class 282, 538
onNotReady 485, 584
onOffHook 422, 571
onOnHook 422, 571
onOpened 463, 580
onOutOfService 449, 577
onParking 423, 572
onPartyAdded 423, 572
ChatConnection class 548
ChatConnection class callback event methods 320
onPartyChanged 423, 572
ChatConnection class 548
ChatConnection class callback event methods 321
onPartyRemoved 423, 572
ChatConnection class 548
ChatConnection class callback event methods 321
onPresenceChanged 485, 584
onProcessing
EmailConnection class callback event methods 343
GenericConnection class callback event methods 372
onProperties
ChatConnection class 548
ChatConnection class callback event methods 321
onPushURL
ChatConnection class 548
ChatConnection class callback event methods 321
onReady 485, 584
onReconnecting 424, 572
onRegistered 485, 584
onRegistering 486, 584
onRejected 424, 572
ChatConnection class 548
ChatConnection class callback event methods 322
EmailConnection class 555
EmailConnection class callback event methods 344
GenericConnection class 562
GenericConnection class callback event methods 372
onRejecting 424, 572
onReleased
ChatConnection class 548
ChatConnection class callback event methods 322
onReleasing 424, 572
ChatConnection class 548
ChatConnection class callback event methods 322
onRestart 449, 578
onRetrieving 425, 572


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

onRevoked
ChatConnection class 548
ChatConnection class callback event methods 322
EmailConnection class 555
EmailConnection class callback event methods 344
GenericConnection class 562
GenericConnection class callback event methods 373
onRinging 425, 572
onSetcallresult 425, 572
onSetcallresultDNC 425, 572
onSetcallresultReschedule 426
onSettingDnd 356, 557
onSettingNotReady 486, 584
onSettingPresence 486, 584
onSettingReady 486, 584
onSettingWorkNotReady 487, 584
onSettingWorkReady 487, 584
onStat 566, 580
Group class 388, 566
Task class 466
User class 487, 584
onStat1
Buddy class 291, 540
Group class 389, 566
User class 487
onStat1(
User class 584
onStat2
Buddy class 292, 541
Group class 389, 566
User class 488, 584
onState 292, 541
onTalking 426, 572
ChatConnection class 548
ChatConnection class callback event methods 323
onTaskAdded 566
__UQAddress class 269
_UQAddress class 534
Group class 389
onTaskRemoved
__UQAddress class 270
Group class 389
onTransfering 426, 572
onTyping
ChatConnection class 548
ChatConnection class callback event methods 323
onUnassigned 270
_UQAddress class 535
onUnknown 488, 584
onUnmuted 426, 572
onUnregistered 488, 584
onUnregistering 488, 584
onUp 446, 577
onUpdatingCD 427, 572
onUserdataChanged
ChatConnection class 548
ChatConnection class callback event methods 323
EmailConnection class 555
EmailConnection class callback event methods 344
GenericConnection class 562
GenericConnection class callback event methods 373
onUserDataChanged 427, 572
onUserRegistered 463, 580
onUserUnregistered 463, 580
onWithdraw_REQ
EmailConnection class callback event methods 344
GenericConnection class callback event methods 373
onWorkNotReady 489, 584
onWorkReady 489, 584
open 454, 579
openSession 441, 575
Outbound Call page 48
outbound call sample page 47
overflow
administering overflow tasks 148
checking for overflowed persistent tasks 116
specifying task overflow default timeout 105
Overflow Administration component (MCF_OVERFLOWL_CMP) 148
Overflow Tasks page 148
overview
MultiChannel Framework 1

# P

parameters 230
park 411, 570
PeopleCode
sample page 529
using built-in functions 201
PeopleCode broadcast
configuring 166
PeopleCode Broadcast page 166
PeopleCode Sample page 529
PeopleSoft CTI *See* CTI
PeopleSoft Integration Broker *See* Integration Broker
PeopleSoft MultiChannel API (PSMCAPI) *See* PSMCAPI, PSMCAPI
percentIdleTime
AgentStatistics class 284, 538
percentTimeAvailable
AgentStatistics class 284, 538
percentTimeInCurrentState
AgentStatistics class 285, 538
percentTimeUnavailable
AgentStatistics class 285, 538
performance issues
clustering REN servers 67
recording agent state information 139
restarting MCF servers 94
setting CTI agent tracing 40
tuning MCF clusters 108
permission lists, REN server 69
personalization 41
chat 127
CTI pop-ups 60
dialing phone numbers on pages 48
floating CTI Console 41
Personalization page (CTI) 41
Personalize Chat page 127
Personalize Presence page 122
phone books 40
using CTI shared 25


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

physical queues
active/inactive 99
administering 133
assigning agents 121
balancing agents/tasks across 136
deleting 98
displaying number of unassigned tasks/logged agents 110
forwarding chats 155, 234
implementing scalability 93
moving 135
moving agents 134
selecting MCF clusters 99
understanding *See Also* queues
viewing MCF cluster IDs 99
viewing physical queue IDs 99
viewing state summary 138
Physical Queues component (MCF_ACCPT_TASK_CMP) 133
ping tests, REN server 83, 96
pop-ups 42
configuring 44
notifying CTI agent of calls 60
selecting pop-up mode for task/chat acceptance windows 127
setting up Cisco 45
setting up Genesys 45
using signon screens 26
POP3 *See* Post Office Protocol 3 (POP3)
Post Office Protocol 3 (POP3)
calculating email size 512
setting the connector threshold 509
presence, agents 227
presences 272, 535
presenceText 497, 587
primary queue servers 96
priority 466, 580
PSADMIN
configuring REN servers 75
setting application server tracing 147
PSMCAPI 16
components 18
configuring 18
CTI architecture 15
file locations 18
security 20
using older adapters 18
PSMCAPI certificates 595
PSMC class
constructor 435, 575
fields *See Also* PSMC class fields, 575
method 437
methods 575
PSMC class fields
renserver 436
servers 437
sessions 437
PSMC class methods
closeSession 437
getCallById 438
getChatById 438
getEmailById 439
getGenericTaskById 439
getLineById 440
openSession 441
start 441
stop 442
PSMC constructor 436, 575
PSMC object 575
PT_CTI_AGENT component 36
PT_CTI_DEMOOUTB component 47, 178
PT_CTI_MISC component 22
PT_CTI_QUEUE component 36
pushURL
ChatConnection class 547
ChatConnection class methods 314

# Q

question
A2AChat class 278
AppData class 288, 539
Chat class 301, 544
Email class 330, 551
GenericTask class 379, 564
queue-based broadcast 159
queue canned messages 218
queue canned URL 219
Queue Configuration component (PT_CTI_QUEUE) 36
Queue Configuration page 37
queuedWaitTime 567
GroupStatistics class fields 392
queue IDs 218
logical 120
physical 121
understanding character requirements 98
viewing physical queue IDs 99
Queue page 98
queues
active/inactive 218
configuring 37
defining canned URL 219
defining for third party 217
defining logical queues
*See Also* logical queues
deleting 218
MCF *See Also* MCF queues
monitoring 197
physical *See Also* physical queues
task queues, refreshing 115
universal 7
using queue servers *See Also* queue servers
viewing cluster IDs 218
queue server
state 116
queue servers
assigning tasks to agents *See Also* tasks
changing for MCF clusters 96
configuring MCF clusters 95
defining failover 92
implementing scalability 93
logging the status 111
master/slave
*See Also* master and slave servers
optimizing performance 109
primary/backup within MCF clusters 96
setting maximum agent timeouts 115
specifying interval for clearing deleted tasks from memory 115
viewing the server state 137
Queue Server State component (MCF_QSERVSTATE_CMP) 137
Queue Server State page 137

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

Queue State Summary component (MCF_QUEUESTATE_CMP) 137
Queue State Summary page 138
queue statistics
monitoring 110
queue statistics, monitoring 197
queueTime 298, 543, 581
TaskStatistics class fields 468
queueUpTime 567
GroupStatistics class fields 392

# R

real-time event notification (REN) servers *See* REN servers
reason 434, 574
Reason class
constructor 442, 576
fields *See Also* Reason class fields, 576
Reason class fields
code 443
desc 443
reasonData1 443
reasonData2 444
reasonData3 444
Reason Code page 27
Reason constructor 442, 576
reasonData1 443, 576
reasonData2 444, 576
reasonData3 444, 576
reasonFlag 568
GroupStatistics1 class 395
UserStatistics1 class 497, 587
Reason object 576
recentTask 397, 568
reconnect 411, 570
reeperinterval 115
register 476, 583
registerAddress 454, 579
registerBuddiesBulk 455, 456
registerBuddy 455, 579
registered 387
Group class 566
registeredAddresses 471, 582
registerGroup 456, 579
registerUser 457, 579
reject 412, 570
ChatConnection class 547
ChatConnection class methods 314
EmailConnection class 554
EmailConnection class methods 340
GenericConnection class 561
GenericConnection class methods 369
relativeQueueLoad 567
GroupStatistics class fields 392
release 412, 570
ChatConnection class 547
ChatConnection class methods 315
REN_CLUSTER_CMP component 80
REN_SERVER_CMP component 71
REN Cluster component (REN_CLUSTER_CMP) 80
REN Java client
importing certificates 592
REN Permissions page 69
renserver 436
renServer 575
REN server
configuring in PSADMIN 75
security 20
RenServer class
callback event methods 576
constructor 444, 576
event method
*See Also* RenServer class callback event methods
fields *See Also* RenServer class fields, 576
RenServer class callback event methods
onDown 445
onUp 446
RenServer class fields
isRunning 445
url 445
REN Server Cluster Members page 84
REN Server Cluster Owner page 83
REN Server Cluster page 80
REN Server Cluster Root Path 82
REN Server Cluster URL 82
REN Server component (REN_SERVER_CMP) 71
REN Server Configuration page 77
RenServer constructor 444, 576
RenServer object 576
REN servers
clustering 80
configuring 71, 74, 94
configuring cluster members 84
configuring clusters 80
configuring failover 65
configuring MCF clusters 95
configuring ownership 83
configuring reverse proxy *See Also* RPS
configuring scalability 65
configuring security 65, 69
configuring SSL-enabled clusters 80
creating 74
defining 77
defining permission lists 69
determining configuration options 71
logging 112, 213
performing buffer/ping tests 82, 96
understanding 65
understanding clusters 67
understanding SSL-enabled 68
retrieve 413, 570
reverse proxy server *See* RPS
reverse proxy servers
configuring REN server security 69
configuring REN servers with SSL 73
setting REN server browser URLs 83
routing, third party 204
RPS
configuring 84
configuring Apache-based 87
configuring WebLogic 8.1 85
understanding configuration 85

# S

Sample Console page 240
Sample Monitor - Agent States 195
Sample Monitor - Queue Statistics 197
sample pages


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

email 516
instant messaging 528
instant messaging button 530
Monitor Agents page 195
PeopleCode 529
Sample Monitor - Agent States 195
using customer chat 170
using Email page 176
using Email window 177
using Generic Event page 173
using Generic Event window 175
using Monitor Queues page 197
using Sample Monitor - Queue Statistics 197
using URL wizard 172
working with 169
sample pages, third party 239
scalability
configuring MCF 94
configuring REN server clusters 80
configuring REN servers 65
understanding 89
using queue servers 93
Secure Socket Layer (SSL)
configuring REN/reverse proxy servers 73
configuring REN server clusters 67
configuring REN server security 69
using the MultiChannel Console 152
security
accessing CTI 53
applying CTI applet protection 21
configuring REN servers 65, 69
opening MultiChannel consoles 152
PSMCAPI 20
REN server 20
setting security zones for web/REN servers 77, 152
setting the authentication domain for REN servers 74
specifying internal/external URLs for MCF clusters 95
sendDTMF 413, 570
Server.TYPE_CTI 382
Server.TYPE_UQ 382, 565
Server.TYPE.CTI 565
server authentication 68
Server class
callback event methods 577
constructor 446, 577
event method
*See Also* Server class callback event methods
fields *See Also* Server class fields, 577
Server class callback event methods
onBroadcast 448
onHbLost 448
onHbRecovered 449
onInService 449
onOutOfService 449
onRestart 449
Server class fields
id 447
info 447
state 447
type 447
Server constructor 446, 577
serverId 452
serverID 578
Server object 577
servers 437, 575
application *See Also* application servers
configuring reverse proxy *See Also* RPS
instant messaging *See Also* instant messaging
mail *See Also* mail servers
master/slave
*See Also* master and slave servers
MCF *See Also* MCF servers
queue *See Also* queue servers
REN 8
understanding REN servers
*See Also* REN servers
understanding web servers
*See Also* REN servers
using MCF log *See Also* MCF log servers
service request masks 36
Session class
callback event methods 579
constructor 450, 578
event method
*See Also* Session class callback event methods
fields *See Also* Session class fields, 578
method 452
methods 578
Session class callback event methods
onAddressRegistered 461
onAddressUnregistered 461
onBuddyRegistered 461
onBuddyUnregistered 461
onClosed 462
onError 462
onGroupRegistered 462
onGroupUnregistered 462
onInfo 463
onOpened 463
onUserRegistered 463
onUserUnregistered 463
Session class fields
addresses 450
buddies 451
groups 451
id 451
intervalBetweenReqs 451
serverId 452
state 452
user 452
Session class methods
broadcastSubscribe 453
broadcastUnsubscribe 453
close 453
open 454
registerAddress 454
registerBuddiesBulk 455, 456
registerBuddy 455
registerGroup 456
registerUser 457
setAutoRecovery 457
statPublish 458
unregisterAddress 458
unregisterBuddy 459
unregisterGroup 459
unregisterUser 460
Session constructor 450, 578
Session object 578
sessions 437, 575
setAutoRecovery 457, 579
setcallresult 414, 570


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

setcallresultDNC 414, 570
setcallresultReschedule 415, 570
setDnd 353, 557
setNotReady 477, 583
setPresence 478, 583
setPresenceUq 478, 583
setReady 479, 583
setWorkNotReady 583
setWorkReady 480, 583
Shared Phone Book page 25
Siemens Hicom 300 31
signon, single *See* single signon
Simple Mail Transfer Protocol (SMTP) 510
single signon
    accessing REN servers 69
    setting the default URL 26
    supporting on CTI 47
slave servers *See* master and slave servers
SMTP 510
socket binding 77
SSL *See* Secure Socket Layer (SSL)
SSL only 78
SSL Port 78
ST_LOGGEDIN 273, 535
ST_LOGGEDOUT 273, 535
ST_NOTREADY 273, 535
ST_READY 273, 535
ST_UNKNOWN 273, 535
ST_WORKNOTREADY 273, 535
ST_WORKREADY 274, 535
start 441, 575
statdump 116
state
    ChatConnection class 546
    ChatConnection class fields 309
    EmailConnection class 554
    EmailConnection class fields 336
    GenericConnection class 561
    GenericConnection class fields 365
    Line class 400, 569
    Server class 447, 577
    Session class 452, 578
    UserStatistics1 class 497, 587
states 272, 535
statesUq 471, 582
static push URLs
    defining 100
    defining agent-specific 128
statistics
    _User class 274, 535
    Call class 294, 541
    Chat class 301, 544
    Email class 330, 551
    GenericTask class 379
    Group class 387, 566
statistics, logging 113, 214
statistics1
    _User class 274, 536
    Group class 388, 566
statistics2
    _User class 274, 536
    Group class 388, 566
statPublish 458, 579
status
    detecting instant messaging presence 530, 531
    dialing outbound calls 62
    logging for agents 111
    logging for queue servers 111
    switching Agent Ready status 61
    understanding hold status 61
    viewing for CTI agents 56
stop 442, 575
strData 288, 539
subject
    A2AChat class 278
    AppData class 288, 539
    Chat class 301, 544
    Email class 330, 552
    GenericTask class 380, 564
system-wide broadcast 159

# T

talkTime 298, 543, 581
    TaskStatistics class fields 469
task
    configuring for third party 231
Task.TYPE_A2ACHAT 383, 565
Task.TYPE_CHAT 383, 565
Task.TYPE_CTI 383, 565
Task.TYPE_EMAIL 383, 565
Task.TYPE_GENERIC 384, 565
taskAcceptedCurrentLogin
    AgentStatistics class 538
Task class
    constructor 464, 580
    fields *See Also* Task class fields, 580
    Hierarchy 464
Task class fields
    caseid 465
    cost 465
    customerid 465
    group 466
    id 466
    onStat 466
    priority 466
    type 466
    urlAbs 467
    urlRel 467
Task constructor 464, 580
Task object 580
task queues, refreshing 115
tasks
    _UQAddress class 534
    A2AChatAddress class 281, 537
    activating 152
    active/inactive physical queues 99, 218
    administering escalation/overflow 148
    ChatAddress class 545
    checking for expired/overflowed persistent tasks 116
    configuring 101
    delivering/assigning 8
    displaying number of unassigned tasks per queue 110
    managing via MultiChannel Console 151
    managing with MultiChannel Console 8
    moving agents 134
    moving queues associated with agents 135
    refreshing the task queue 115
    selecting acceptance mode in chat 127
    selecting task assignment actions 130
    setting agent skill level 121
    specifying chat greeting interval 106


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

specifying default acceptance timeout 105
specifying default cost 103
specifying default escalation/overflow timeout 105
specifying default priority/agent skill level 104
specifying how many/often to read persistent tasks 112, 114
specifying interval for clearing deleted tasks from memory 115
specifying range for parameter values 106
tuning clusters 108
universal queue 7
using PeopleCode built-in functions 201
viewing the enqueued task count 106
Tasks field 267
TaskStatistics class
constructor 467, 581
fields
*See Also* TaskStatistics class fields, 581
TaskStatistics class fields
data 468
holdTime 468
queueTime 468
talkTime 469
TaskStatistics constructor 581
TaskStatistics object 581
taskTotalTimeInSystem 396, 568
TCP_NODELAY 77
teleset 59
third-party chat 232
third-party flag 209
third party
configuring MCF 208
configuring task 231
creating agents 222
defining agents 220
defining MCF cluster page 210
defining outing rules 207
defining queues 217
defining system requirements 207
defining the flag 209
notifying clusters 215
setting up buddy list 223
specifying agent's media 228
specifying languages that agent supports 229
tuning MCF cluster parameters 212
understanding routing 204
using Reason Code page 27
viewing broadcast logs 238
viewing event logs 236
working with sample pages 239
timeCurrentLogin
AgentStatistics class 285, 538
timeElapsedOldestTask 398, 567, 568
GroupStatistics class fields 392
timeElapsedRecentTask 398, 568
timeIdle 499, 588
timeInCurrentState 497, 499, 587, 588
timeNotReady 499, 588
timeouts
setting maximum agent timeouts for queue server logout 115
specifying for chats 106
specifying the default for task acceptance 105
task escalation/overflow 105, 148
timeSinceLoggedIn 498, 587
timeSinceLogin 499, 588
timeSinceStart 396, 568
timeWorking
AgentStatistics class 285, 538
time zone offsets 502
timinginterval 116
totalTaskAcceptedLogin
AgentStatistics class 285, 538
totalTaskDoneLogin
AgentStatistics class 286, 539
totalTaskUnassignedLogin
AgentStatistics class 286
totalTimeAvailable 500, 588
totalTimeUnavailable 500, 588
trace 502
tracing
setting for agents 130
setting for CTI agents 40
using application servers 147
transfer 416, 571
transfer conference setup masks 36
transferMute 416, 571
transferring callers 58
troubleshooting CTI errors 40
tuning clusters 212
type
A2AChat class 278, 536
MediaType class 435, 574
Server class 447, 577
Task class 466, 581
typing
ChatConnection class 547
ChatConnection class methods 315


# U


unavailableDuration 286, 539
uniqueId 288, 539
uniqueueId 279, 537
universal queue 7
universal queue classes 202
unmute 417, 571
unregister 481
unRegister 583
UserData class 586
unregisterAddress 458, 579
unregisterBuddy 459, 579
unregisterGroup 459, 579
unregisterUser 460, 579
updateCallData 417, 571
UQ Address constructor 266
UQ Cluster (universal queue cluster) page 94
url 539, 576
AppData class 289
RenServer class 445
urlAbs 467, 581
urlRel 467, 581
URLs
defining agent-specific 128
defining REN server cluster 67
defining static push 100
grabbing 172
sending/grabbing via chat 155, 234
specifying internal/external URLs for MCF clusters 95
using the URL wizard 172


Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

viewing details for MCF clusters 107
URL Wizard page 172
user 574
MCEvent class 434
Session class 452, 578
UserCaps class
constructor 489, 490, 585
fields 585
UserCaps class fields
canHandleChat 490
canHandleEmail 490
canHandleGeneric 490
canHandleVoice 491
canLogin 491
canLogout 491
canRefreshState 491
canSetNotReady 491
canSetPresence 492
canSetReady 492
canSetWorkNotReady 492
canSetWorkReady 492
UserCaps constructor 489, 585
UserCaps object 585
User class
callback event methods 583
constructor 469, 582
event method
*See Also* User class callback event methods
fields *See Also* User class fields, 582
method 471
methods 583
User class callback event methods
onCapabilitiesChanged 481
onCtiBusy 482
onCTIBUSYUq 482
onCtiClear 482
onDropped 482
onError 483
onInfo 483
onLoggedIn 483
onLoggedOut 483
onLoggingIn 484
onLoggingOut 484
onMediaDisabled 484
onMediaEnabled 484
onNotReady 485
onPresenceChanged 485
onReady 485
onRegistered 485
onRegistering 486
onSettingNotReady 486
onSettingPresence 486
onSettingReady 486
onSettingWorkNotReady 487
onSettingWorkReady 487
onStat 487
onStat1 487
onStat2 488
onUnknown 488
onUnregistered 488
onUnregistering 488
onWorkNotReady 489
onWorkReady 489
User class fields
addresses 470
agentPassword 471
language 471
registeredAddresses 471
statesUq 471
User class methods
ctiBusyUq 472
disableMedia 472
enableMedia 473
enableMediaregister 476
enableMediasetNotReady 477
isMediaEnabled 474
login 474
loginUq 475
logout 475
logoutUq 476
setPresence 478
setPresenceUq 478
setReady 479
setWorkNotReady 479
setWorkReady 480
unregister 481
User constructor 469, 582
userdata
GenericTask class 380
userData 542
Chat class 302, 544
Email class 330, 552
GenericTask class 564
UserData class
constructor 492, 585
fields 493, 495
methods *See Also* UserData class methods, 586
UserData class field constant 493
UserData class methods
addKeyValue 494
UserData constructor 493, 586
UserData object 586
userId 289, 539
username 289, 539
User object 582
UserStatistics1 class
constructor 494, 586
fields 587
UserStatistics1 class fields
availableCost 495
ctiBusy 495
currentQueue 496
mostRecentTaskData 496
mostRecentTaskId 496
numTaskAccepted 496
numTasksDone 496
numTasksUnassigned 497
presenceText 497
reasonFlag 497
state 497
timeInCurrentState 497
timeSinceLoggedIn 498
UserStatistics1 constructor 495, 587
UserStatistics1 object 587
UserStatistics2 class
constructor 498, 587
fields 498, 587
UserStatistics2 class fields
currentQueue 499
timeIdle 499
timeInCurrentState 499
timeNotReady 499
timeSinceLogin 499
totalTimeAvailable 500
totalTimeUnavailable 500
UserStatistics2 constructor 498, 587



Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.

UserStatistics2 object 587

# V

virus scan
    enable 512
voice channels *See Also* channels

# W

waitDuration 286, 539
web servers *See* REN servers
Window Config (configuration) page 125
windows
    configuring for agents 125
    configuring pop-ups 44
    personalizing pop-ups 42
    setting up Cisco for pop-ups 45
    setting up Genesys for pop-ups 45
withdraw_RES
    EmailConnection class 554
    EmailConnection class methods 340
    GenericConnection class 561
    GenericConnection class methods 369
wizUrl 289, 539
workload
    implementing queue server scalability 93
    setting maximum for agents 121
    specifying for agents *See Also* tasks
    universal queue 7
wrap-up mode 61
wrap-up time, Genesys 63
wrapup
    ChatConnection class methods 316

# X

XMPP domains 523

# Y

Yahoo! Messenger
    configuring servers 522
    detecting presence 521
    setting the server IP address 523
    updating presence 530

Copyright © 1988, 2010, Oracle and/or its affiliates. All Rights Reserved.