PeopleSoft®

Enterprise PeopleTools 8.46 PeopleBook: PeopleSoft MultiChannel Framework

February 2005

Enterprise PeopleTools 8.46 PeopleBook: PeopleSoft MultiChannel Framework
SKU PT846MCF-B 0205
Copyright © 1988-2005 PeopleSoft, Inc. All rights reserved.

All material contained in this documentation is proprietary and confidential to PeopleSoft, Inc. (“PeopleSoft”), protected by copyright laws and subject to the nondisclosure provisions of the applicable PeopleSoft agreement. No part of this documentation may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, including, but not limited to, electronic, graphic, mechanical, photocopying, recording, or otherwise without the prior written permission of PeopleSoft.

This documentation is subject to change without notice, and PeopleSoft does not warrant that the material contained in this documentation is free of errors. Any errors found in this document should be reported to PeopleSoft in writing.

The copyrighted software that accompanies this document is licensed for use only in strict accordance with the applicable license agreement which should be read carefully as it governs the terms of use of the software and this document, including the disclosure thereof.

PeopleSoft, PeopleTools, PS/nVision, PeopleCode, PeopleBooks, PeopleTalk, and Vantive are registered trademarks, and Pure Internet Architecture, Intelligent Context Manager, and The Real-Time Enterprise are trademarks of PeopleSoft, Inc. All other company and product names may be trademarks of their respective owners. The information contained herein is subject to change without notice.

Open Source Disclosure

PeopleSoft takes no responsibility for its use or distribution of any open source or shareware software or documentation and disclaims any and all liability or damages resulting from use of said software or documentation. The following open source software may be used in PeopleSoft products and the following disclaimers are provided.

Apache Software Foundation

This product includes software developed by the Apache Software Foundation (http://www.apache.org/). Copyright © 1999-2000 The Apache Software Foundation. All rights reserved.

THIS SOFTWARE IS PROVIDED “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

OpenSSL

Copyright © 1998-2003 The OpenSSL Project. All rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

SSLeay

Copyright © 1995-1998 Eric Young. All rights reserved.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com). Copyright © 1995-1998 Eric Young. All rights reserved. THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Loki Library

Copyright © 2001 by Andrei Alexandrescu. This code accompanies the book:

Alexandrescu, Andrei. "Modern C++ Design: Generic Programming and Design Patterns Applied." Copyright © 2001 Addison-Wesley. Permission to use, copy, modify, distribute and sell this software for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

Helma Project

Copyright © 1999-2004 Helma Project. All rights reserved.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE HELMA PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Helma includes third party software released under different specific license terms. See the licenses directory in the Helma distribution for a list of these license.

Sarissa

Copyright © 2004 Manos Batsis.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

Contents

General Preface

About This PeopleBook xxvii

PeopleSoft Application Prerequisites xxvii

PeopleSoft Application Fundamentals xxvii

Documentation Updates and Printed Documentation xxviii

Obtaining Documentation Updates xxviii

Ordering Printed Documentation xxviii

Additional Resources xxix

Typographical Conventions and Visual Cues xxx

Typographical Conventions xxx

Visual Cues xxxi

Country, Region, and Industry Identifiers xxxii

Currency Codes xxxii

Comments and Suggestions xxxii

Common Elements Used in PeopleBooks xxxiii

Preface

PeopleSoft MultiChannel Framework Preface xxxv

PeopleSoft MultiChannel Framework xxxv

Additional Resources xxxv

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 3

PeopleSoft MultiChannel Framework 3

PeopleSoft MultiChannel Framework Elements and Channels 3

PeopleSoft MultiChannel Elements 4

PeopleSoft MultiChannel Framework Channels 4

PeopleSoft MultiChannel Framework Universal Queue.....5
PeopleSoft MultiChannel Console.....6
PeopleSoft MultiChannel Framework Architecture.....6
PeopleSoft MultiChannel Framework Server Architecture.....6
Chat Architecture.....6
Email Architecture.....8

Chapter 3
Configuring PeopleSoft CTI.....11
Understanding PeopleSoft CTI.....11
PeopleSoft CTI.....11
PeopleSoft CTI Components.....12
Understanding PeopleSoft CTI Components.....13
PeopleSoft CTI Requirements.....14
PeopleSoft MultiChannel API.....14
Required Security for PSMCAPI.....14
JavaScript MultiChannel API.....15
Genesys Components.....15
Cisco Components.....16
Java Runtime Environment.....16
Configuring PeopleSoft CTI.....16
Pages Used to Configure CTI.....17
Installing PeopleSoft CTI.....18
Enabling PeopleSoft CTI.....18
Creating CTI Configurations.....19
Configuring the CTI Non-Applet Console.....19
Configuring the CTI Applet Console.....20
Using the Shared Phone Book Page.....22
CTI Genesys Page.....23
CTI Cisco Page.....24
Miscellaneous Page.....26
Configuring Queues and Agents.....27
Pages Used to Configure Queues and Agents.....28
Configuring CTI Queues.....28
Configuring CTI Agents.....28
Using the Phone Book Page.....30
Using the Personalization Page.....31
Viewing Information on the Agent Information Page.....32
Using Other CTI Options.....32

Configuring Pop-up Windows....32
Supporting Single Signon....35
CTI Event Logging....35
Implementing "Free Seating"....35
Using the CTI Sample Pages....35
Page Used to Demonstrate Outbound Calls....36
Using the Outbound Call Page....36

Chapter 4
Using PeopleSoft CTI....37
Understanding PeopleSoft CTI....37
PeopleSoft CTI....37
Cisco Switch Considerations....38
Using PeopleSoft CTI....38
Getting Started....39
Using the CTI Console....40
Selecting Call Actions....41
Answering a Call....42
Transferring a Caller....42
Initiating Conference Calls....44
Working with the Hold Status....44
Disconnecting a Caller....45
Switching Agent Ready Status....45
Dialing an Outbound Call....45
Completing a Call....46
Using Hot Keys....46

Chapter 5
Configuring REN Servers....49
Understanding REN Servers....49
REN Server Failover, Scalability, and Security Configuration....49
REN Server Failover....50
REN Server Clusters....51
Configuring REN Server Security....51
Understanding REN Server Security Configuration....51
Defining Permission Lists for REN Server Access....52
Configuring REN Servers....53
Selecting REN Server Configuration Options....53

Page Used to Configure REN Servers....55
Configuring REN Servers....55
Defining REN Servers....58
Configuring REN Server Clusters....58
Pages Used to Configure REN Server Clusters....59
Clustering REN Servers....59
Specifying REN Server Ownership....61
Specifying REN Server Cluster Members....61
Example: Configuring a Reverse Proxy Server with a REN Server....62
Understanding RPS Configuration....62
Example: Configuring a WebLogic 8.1 RPS for a REN Server on Another Host Machine....62

Chapter 6
Configuring MCF Clusters....65
Understanding PeopleSoft MultiChannel Framework Scalability and Failover....65
PeopleSoft MultiChannel Framework Cluster Architecture....65
MCF Clusters....66
Logical Queues and Physical Queues....67
Queue Server Failover....67
MCF Log Server Failover....68
MCF Cluster Configuration....68
Queue Server Scalability....69
Recommended Configurations....69
Configuring MCF Servers....69
Understanding MCF Server Configuration....69
Creating Queue Servers....70
Configuring MCF Clusters....70
Page Used to Configure MCF Clusters....70
Defining MCF Clusters....70

Chapter 7
Configuring MCF Queues....73
Defining Queues....73
Pages Used to Define Queues....74
Defining Queues....74
Defining Chat Responses....75
Defining Static Push URLs....76
Configuring Tasks....77

Page Used to Configure Tasks....77
Configuring Tasks....77
Viewing the Cluster Summary....81
Page Used to View the Cluster Summary....81
Viewing the Cluster Summary....81
Tuning Cluster Parameters....81
Page Used to Tune Clusters....82
Tuning Cluster Parameters....82
Notifying Clusters of Changed Parameters....86
Page Used to Notify Clusters of Changed Parameters....86
Notifying Clusters of Changed Parameters....86

Chapter 8
Configuring MCF Agents....89
Defining Agents....89
Pages Used to Define Agents....89
Creating Agents....89
Specifying Languages That an Agent Supports....91
Personalizing an Agent's Presence....91
Defining Optional Agent Characteristics....92
Pages Used to Define Optional Agent Characteristics....93
Setting Up Buddy Lists....93
Configuring Windows....94
Personalizing Chat....95
Specifying Agent-Specific URLs....96
Specifying Miscellaneous Parameters....97

Chapter 9
Administering PeopleSoft MultiChannel Framework....99
Administering Physical Queues....99
Pages Used to Administer Physical Queues....100
Moving Agents Between Physical Queues....100
Moving Queues....101
Balancing Queues....102
Viewing Queue Server, Queue, and Agent States....102
Pages Used to View Queue Server, Queue, and Agent States....103
Viewing the Queue Server State....103
Viewing the Queue State Summary....104

Viewing the Agent State Summary....104
Viewing Chat and Event Logs....105
Pages Used to View Chat and Event Logs....106
Viewing Chat Logs....106
Viewing Event Logs....107
Viewing MCF Logs....110
Administering Overflow and Escalated Tasks....111
Pages Used to Administer Overflow and Escalated Tasks....111
Administering Overflow Tasks....111
Administering Escalated Tasks....112

Chapter 10
Managing Tasks and Using Chat in PeopleSoft MultiChannel Framework....113
Managing Tasks with the MultiChannel Console....113
Using the MultiChannel Console to Work with Tasks....113
Communicating with Customers and Agents Using Chat....115
Using the Agent Chat Window....115
Using the Customer Chat Window....117

Chapter 11
Developing for PeopleSoft MultiChannel Framework....121
Working with Sample Pages....121
Pages Used to Work with Sample Pages....122
Using the Customer Chat Sample Page....122
Using the URL Wizard....123
Using the Generic Event Sample Page....125
Using the Generic Event Window....126
Using the Email Sample Page....127
Using the Email Window....128
Using and Demonstrating JSMCAPI....129
Understanding JSMCAPI....129
Common Elements Used in this Section....130
Pages Used to Demonstrate JSMCAPI....130
Using the CTI Sample Console....131
Using the Agent Console Page....135
Using the Monitor Agents Page and Sample Monitor - Agent States Page....137
Using the Monitor Queues Page and Sample Monitor - Queue Statistics Page....139
Using PeopleCode Built-in Functions with PeopleSoft MultiChannel Framework....142

Using Universal Queue Classes.....143

Chapter 12
JSMCAPI Classes.....145
Understanding JSMCAPI.....145
_Address Class Hierarchy.....145
_Address Class Constructor.....146
_Address.....147
_Address Class Fields.....147
caps.....147
id.....147
_Address Class Callback Event Method.....147
onError.....147
_UQAddress Class Constructor.....148
_UQAddress.....148
_UQAddress Class Fields.....148
Tasks.....148
_UQAddress Methods.....148
acceptTask.....149
dequeueTask.....149
_UQAddress Class Callback Event Methods.....149
onAccepted.....150
onAcceptingTask.....150
onDequeueingTask.....150
onNotify.....150
onTaskAdded.....150
onTaskRemoved.....150
onUnassigned.....151
_User Class Hierarchy.....151
_User Class Constructor.....151
_User.....151
_User Class Fields.....152
agentID.....152
caps.....152
id.....152
name.....152
presences.....153
states.....153
ST_LOGGEDIN.....153

ST_LOGGEDOUT.....153
ST_NOTREADY.....153
ST_READY.....153
ST_UNKNOWN.....153
ST_WORKNOTREADY.....153
ST_WORKREADY.....154
statistics.....154
statistics1.....154
statistics2.....154
A2AChat Class Constructor.....154
A2AChat.....154
A2AChat Class Fields.....155
address.....155
agentID.....155
agentName.....155
appData.....155
chatType.....156
id.....156
jr.....156
type.....156
uniqueueId.....157
A2AChat Class Method.....157
getURL.....157
A2AChatAddress Class Constructor.....157
A2AChatAddress.....157
A2AChatAddress Class Fields.....158
id.....158
tasks.....158
A2AChatAddress Class Method.....158
initiateChat.....158
A2AChatAddress Class Callback Event Methods.....159
onChatEnded.....159
onInitiatingChat.....159
onNotify.....159
AgentStatistics Class Constructor.....160
AgentStatistics.....160
AgentStatistics Class Fields.....160
averageCallDuration.....160
averageHoldDuration.....160
callsHandled.....160

data....161
unavailableDuration....161
waitDuration....161
AppData Class Constructor....161
AppData....161
AppData Class Fields....161
data....162
groupId....162
jr....162
question....162
strData....162
subject....162
uniqueId....162
url....163
userId....163
username....163
wizUrl....163
AppData Class Method....163
addKeyValue....163
Buddy Class Constructor....164
Buddy....164
Buddy Class Fields....164
Buddy Class Callback Event Methods....165
onStat1....165
onStat2....165
onState....165
Call Class Constructor....165
Call....166
Call Class Fields....166
line....166
statistics....167
CallData Class Constructor....167
CallData....167
CallData Class Fields....167
ani....167
callId....167
callType....168
data....168
dnis....168
CallData Class Method....168

addKeyValue....168
CallStatistics Class Constructor....169
CallStatistics....169
CallStatistics Class Fields....169
data....169
holdTime....169
queueTime....169
talkTime....170
Chat Class Constructor....170
Chat....170
Chat Class Fields....170
address....171
agentId....171
appData....171
customerName....171
groupId....171
question....172
subject....172
Chat Class Method....172
getUrl....172
ChatAddress Class Constructor....172
ChatAddress....173
ChatAddress Class Fields....173
ChatAddress Class Methods....173
ChatAddress Callback Event Methods....174
Email Class Constructor....174
Email....174
Email Class Fields....175
agentId....175
appData....175
Email Class Method....175
getUrl....175
EmailAddress Class Constructor....176
EmailAddress....176
EmailAddress Class Fields....176
EmailAddress Class Methods....177
EmailAddress Callback Event Methods....177
Extension Class Constructor....177
Extension....177
Extension Class Fields....178

forwardMode....178
isDnd....178
lines....178
numOfLines....179
Extension Class Methods....179
cancelDnd....179
cancelForwardSet....179
forwardSet....180
getDialingLine....180
getFreeLine....180
getLineByConnectionId....181
getOffHookLine....181
setDnd....181
Extension Class Callback Event Methods....182
onCancelingDnd....182
onCancelingForward....182
onDnd....182
onForwardCanceled....182
onForwarded....183
onForwarding....183
onSettingDnd....183
ExtensionCaps Class Constructor....183
ExtensionCaps....183
ExtensionCaps Class Fields....184
canCancelDnd....184
canDial....184
canFwdBusy....184
canFwdBusyNoAnswer....184
canFwdCancelForward....184
canFwdDefault....184
canFwdNoAnswer....185
canFwdUnconditional....185
canRefreshState....185
canSetDnd....185
ForwardMode Class Constructor....185
ForwardMode....185
ForwardMode Class Field....186
mode....186
GenericAddress Class Constructor....186
GenericAddress....186

GenericAddress Class Fields....187
GenericAddress Class Methods....187
GenericAddress Class Callback Event Methods....187
GenericTask Class Constructor....188
GenericTask....188
GenericTask Class Fields....188
agentId....189
appData....189
customerName....189
groupId....189
question....189
subject....189
GenericTask Class Method....190
getUrl....190
GLOBALS Class Fields....190
A2AChat.PS_JR....190
A2AChat.TYPE_ANSWER....190
A2AChat.TYPE_CONSULT....191
Server.TYPE_CTI....191
Server.TYPE_UQ....191
Task.TYPE_A2ACHAT....191
Task.TYPE_CHAT....191
Task.TYPE_CTI....192
Task.TYPE_EMAIL....192
Task.TYPE_GENERIC....192
GLOBALS Class Methods....192
GLOBALS class methods....192
isValid....193
Group Class Constructor....193
Group....193
Group Class Fields....193
id....194
name....194
registered....194
statistics....194
statistics1....194
statistics2....194
Group Class Callback Event Methods....195
onStat....195
onStat1....195

onStat2....195
onTaskAdded....195
onTaskRemoved....195
GroupStatistics Constructor....196
GroupStatistics....196
GroupStatistics Fields....196
data....196
numberOfAbandoned....196
numberOfLoggedIn....196
numberOfQueued....197
queuedWaitTime....197
relativeQueueLoad....197
GroupStatistics1 Class Constructor....197
GroupStatistics1....197
GroupStatistics1 Class Fields....197
mostRecentTaskDone....198
mostRecentTaskDoneData....198
mostRecentTaskEnqueued....198
mostRecentTaskEnqueuedData....198
numAgentsAvailable....198
numAgentsLoggedIn....198
numEscalation....198
numOverflow....199
numTaskAccepted....199
numTaskDone....199
numTaskQueued....199
reasonFlag....199
taskTotalTimeInSystem....199
timeSinceStart....199
GroupStatistics2 Class Constructor....200
GroupStatistics2....200
GroupStatistics2 Class Fields....200
averageTaskDuration....200
averageWaitTime....200
oldestTask....200
recentTask....201
timeElapsedOldestTask....201
timeElapsedRecentTask....201
Line Class Constructor....201
Line....201

Line Class Fields....202
call....202
caps....202
connectionid....202
id....202
state....202
Line Class Methods....203
alternate....203
answer....203
attachUserData....204
clear....204
complete....205
conference....205
conferenceSingle....205
dial....206
dropParty....206
getAni....207
getDescr....207
getDnis....207
getReferenceId....208
getUrl....208
grabCall....208
hold....209
park....209
reconnect....210
release....210
retrieve....210
transfer....211
transferMute....211
updateCallData....212
Line Class Callback Event Methods....212
onAlternating....212
onAnswering....212
onAttachingUD....212
onCallDataChanged....213
onCapabilitiesChanged....213
onClearing....213
onCompleting....213
onConferencing....213
onDialing....213

onDropped....214
onError....214
onGrabbing....214
onHeld....214
onHolding....214
onOffHook....215
onOnHook....215
onParking....215
onPartyAdded....215
onPartyChanged....215
onPartyRemoved....215
onReconnecting....216
onReleasing....216
onRetrieving....216
onRinging....216
onTalking....216
onTransfering....217
onUpdatingCD....217
onUserDataChanged....217
LineCaps Class Constructor....217
LineCaps....217
LineCaps Class Fields....218
canAlternate....218
canAnswer....218
canAttachUserDate....218
canClear....218
canComplete....218
canConference....218
canConferenceSingle....219
canDropParty....219
canHold....219
canPark....219
canReconnect....219
canRelease....219
canRetrieve....219
canTransfer....220
canTransferMute....220
canUpdateCallData....220
MCEvent Class Constructor....220
MCEvent....220

MCEvent Class Fields....220
extension....221
group....221
reason....221
user....221
MediaType Class Constructor....221
MediaType....222
MediaType Class Field....222
type....222
PSMC Class Constructor....222
PSMC....222
PSMC Class Fields....223
renserver....223
servers....223
sessions....223
PSMC Class Methods....223
closeSession....224
getCallById....224
getLineById....224
openSession....225
start....225
stop....225
Reason Class Constructor....226
Reason....226
Reason Class Fields....226
code....226
desc....227
RenServer Class Constructor....227
RenServer....227
RenServer Class Fields....227
isRunning....227
url....227
RenServer Class Callback Event Methods....228
onDown....228
onUp....228
Server Class Constructor....228
Server....228
Server Class Fields....229
id....229
info....229

state....229
type....229
Server Class Callback Event Methods....230
onBroadcast....230
onHbLost....230
onHbRecovered....230
onInService....230
onOutOfService....230
onRestart....231
Session Class Constructor....231
Session....231
Session Class Fields....231
addresses....231
buddies....232
groups....232
id....232
serverId....232
state....232
user....232
Session Class Methods....233
close....233
open....233
registerAddress....233
registerBuddy....234
registerGroup....234
registerUser....234
statPublish....235
unregisterAddress....235
unregisterBuddy....236
unregisterGroup....236
unregisterUser....236
Session Class Callback Event Methods....237
onAddressRegistered....237
onAddressUnregistered....237
onBuddyRegistered....237
onBuddyUnregistered....237
onClosed....238
onError....238
onGroupRegistered....238
onGroupUnregistered....238

onInfo....238
onOpened....238
onUserRegistered....239
onUserUnregistered....239
Task Class Hierarchy....239
Task Class Constructor....240
Task....240
Task Class Fields....240
caseid....240
cost....240
customerid....241
group....241
id....241
onStat....241
priority....241
type....241
urlAbs....242
urlRel....242
User Class Constructor....242
User....242
User Class Fields....243
addresses....243
agentPassword....244
language....244
registeredAddresses....244
User Class Methods....244
disableMedia....244
enableMedia....245
isMediaEnabled....245
login....245
loginUq....246
logout....246
logoutUq....247
register....247
setNotReady....248
setPresence....248
setPresenceUq....248
setReady....249
setWorkNotReady....249
setWorkReady....250

unregister....250
User Class Callback Event Methods....251
onCapabilitiesChanged....251
onCtiBusy....251
onCtiClear....251
onDropped....251
onError....252
onInfo....252
onLoggedIn....252
onLoggedOut....252
onLoggingIn....252
onLoggingOut....253
onMediaDisabled....253
onMediaEnabled....253
onNotReady....253
onPresenceChanged....253
onReady....253
onRegistered....254
onRegistering....254
onSettingNotReady....254
onSettingPresence....254
onSettingReady....254
onSettingWorkNotReady....255
onSettingWorkReady....255
onStat....255
onStat1....255
onStat2....255
onUnknown....255
onUnregistered....256
onUnregistering....256
onWorkNotReady....256
onWorkReady....256
UserCaps Class Constructor....256
UserCaps....256
UserCaps Class Fields....257
canLogin....257
canLogout....257
canRefreshState....257
canSetNotReady....257
canSetPresence....258

canSetReady....258
canSetWorkNotReady....258
canSetWorkReady....258
UserData Class Constructor....258
UserData....258
UserData Class Fields....259
UserData Class Field Constants....259
UserData Class Method....259
addKeyValue....259
UserStatistics1 Class Constructor....260
UserStatistics1....260
UserStatistics1 Class Fields....260
availableCost....260
ctiBusy....260
currentQueue....261
mostRecentTaskData....261
mostRecentTaskId....261
numTaskAccepted....261
numTasksDone....261
numTasksUnassigned....261
presenceText....261
reasonFlag....262
state....262
timeInCurrentState....262
timeSinceLoggedIn....262
UserStatistics2 Class Constructor....262
UserStatistics2....262
UserStatistics2 Class Fields....263
currentQueue....263
timeIdle....263
timeInCurrentState....263
timeNotReady....263
timeSinceLogin....263
totalTimeAvailable....263
totalTimeUnavailable....264

Chapter 13
Configuring the Email Channel....265
Understanding the Email Channel....265

Handling Email....266
Configuring PeopleSoft Integration Broker for the Email Channel....266
Pages Used to Configure Integration Broker for Email....267
Configuring the Gateway....267
Configuring GETMAILTARGET Properties....267
Demonstrating the Email Channel....273
Pages Used to Demonstrate the Email Channel....274
Using the GetMail - Server Page....274
Using the MailStore - DB Page....276

Chapter 14
Configuring Instant Messaging....279
Understanding Instant Messaging....279
Configuring Instant Messaging Servers....279
Configuring Instant Messaging....280
Page Used to Configure Instant Messaging....280
Configuring Instant Messaging....280
Using the Instant Messaging Sample Pages....281
Pages Used to Test Instant Messaging....281
Using the PeopleCode Sample Page....281
Using the Button Sample Page....282

Appendix A
JSMCAPI Quick Reference....285
JSMCAPI Classes....285
_Address....285
_UQAddress....285
_User....286
A2AChat....287
A2AChatAddress....288
AgentStatistics....289
AppData....290
Buddy....290
Call....291
CallData....292
CallStatistics....293
Chat....293
ChatAddress....294

Email……295
EmailAddress……296
Extension……297
ExtensionCaps……299
ForwardMode……299
GenericAddress……300
GenericTask……301
GLOBALS……302
Group……303
GroupStatistics……303
GroupStatistics1……304
GroupStatistics2……305
Line……305
LineCaps……308
MCEvent……309
MediaType……309
PSMC……310
Reason……311
RenServer……311
Server……311
Session……312
Task……314
User……315
UserData……318
UserStatistics1……318
UserStatistics2……319

Glossary of PeopleSoft Terms……321

Index ……343

About This PeopleBook

PeopleBooks provide you with the information that you need to implement and use PeopleSoft applications.

This preface discusses:

- PeopleSoft application prerequisites.
- PeopleSoft application fundamentals.
- Documentation updates and printed documentation.
- Additional resources.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common elements in PeopleBooks.

Note. PeopleBooks document only page elements, such as fields and check boxes, that require additional explanation. If a page element is not documented with the process or task in which it is used, then either it requires no additional explanation or it is documented with common elements for the section, chapter, PeopleBook, or product line. Elements that are common to all PeopleSoft applications are defined in this preface.

PeopleSoft Application Prerequisites

To benefit fully from the information that is covered in these books, you should have a basic understanding of how to use PeopleSoft applications.

You might also want to complete at least one PeopleSoft introductory training course, if applicable.

You should be familiar with navigating the system and adding, updating, and deleting information by using PeopleSoft menus, and pages, forms, or windows. You should also be comfortable using the World Wide Web and the Microsoft Windows or Windows NT graphical user interface.

These books do not review navigation and other basics. They present the information that you need to use the system and implement your PeopleSoft applications most effectively.

PeopleSoft Application Fundamentals

Each application PeopleBook provides implementation and processing information for your PeopleSoft applications.

Note. Application fundamentals PeopleBooks are not applicable to the PeopleTools product.

For some applications, additional, essential information describing the setup and design of your system appears in a companion volume of documentation called the application fundamentals PeopleBook. Most PeopleSoft product lines have a version of the application fundamentals PeopleBook. The preface of each PeopleBook identifies the application fundamentals PeopleBooks that are associated with that PeopleBook.

The application fundamentals PeopleBook consists of important topics that apply to many or all PeopleSoft applications across one or more product lines. Whether you are implementing a single application, some combination of applications within the product line, or the entire product line, you should be familiar with the contents of the appropriate application fundamentals PeopleBooks. They provide the starting points for fundamental implementation tasks.

Documentation Updates and Printed Documentation

This section discusses how to:

- Obtain documentation updates.
- Order printed documentation.

Obtaining Documentation Updates

You can find updates and additional documentation for this release, as well as previous releases, on the PeopleSoft Customer Connection website. Through the Documentation section of PeopleSoft Customer Connection, you can download files to add to your PeopleBook Library. You'll find a variety of useful and timely materials, including updates to the full PeopleSoft documentation that is delivered on your PeopleBooks CD-ROM.

Important! Before you upgrade, you must check PeopleSoft Customer Connection for updates to the upgrade instructions. PeopleSoft continually posts updates as the upgrade process is refined.

See Also

PeopleSoft Customer Connection, https://www.peoplesoft.com/corp/en/login.jsp

Ordering Printed Documentation

You can order printed, bound volumes of the complete PeopleSoft documentation that is delivered on your PeopleBooks CD-ROM. PeopleSoft makes printed documentation available for each major release shortly after the software is shipped. Customers and partners can order printed PeopleSoft documentation by using any of these methods:

- Web
- Telephone
- Email

Web

From the Documentation section of the PeopleSoft Customer Connection website, access the PeopleBooks Press website under the Ordering PeopleBooks topic. The PeopleBooks Press website is a joint venture between PeopleSoft and MMA Partners, the book print vendor. Use a credit card, money order, cashier's check, or purchase order to place your order.

Telephone

Contact MMA Partners at 877 588 2525.

Email

Send email to MMA Partners at peoplebookspress@mmapartner.com.

See Also

PeopleSoft Customer Connection, https://www.peoplesoft.com/corp/en/login.jsp

Additional Resources

The following resources are located on the PeopleSoft Customer Connection website:

Resource	Navigation
Application maintenance information	Updates + Fixes
Business process diagrams	Support, Documentation, Business Process Maps
Interactive Services Repository	Interactive Services Repository
Hardware and software requirements	Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation & Software, Hardware and Software Requirements
Installation guides	Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation & Software, Installation Guides and Notes
Integration information	Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation and Software, Pre-built Integrations for PeopleSoft Enterprise and PeopleSoft EnterpriseOne Applications
Minimum technical requirements (MTRs) (EnterpriseOne only)	Implement, Optimize + Upgrade, Implementation Guide, Supported Platforms
PeopleBook documentation updates	Support, Documentation, Documentation Updates
PeopleSoft support policy	Support, Support Policy
Prerelease notes	Support, Documentation, Documentation Updates, Category, Prerelease Notes
Product release roadmap	Support, Roadmaps + Schedules
Release notes	Support, Documentation, Documentation Updates, Category, Release Notes

Resource	Navigation
Release value proposition	Support, Documentation, Documentation Updates, Category, Release Value Proposition
Statement of direction	Support, Documentation, Documentation Updates, Category, Statement of Direction
Troubleshooting information	Support, Troubleshooting
Upgrade documentation	Support, Documentation, Upgrade Documentation and Scripts

Typographical Conventions and Visual Cues

This section discusses:

- Typographical conventions.
- Visual cues.
- Country, region, and industry identifiers.
- Currency codes.

Typographical Conventions

This table contains the typographical conventions that are used in PeopleBooks:

Typographical Convention or Visual Cue	Description
Bold	Indicates PeopleCode function names, business function names, event names, system function names, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call.
Italics	Indicates field values, emphasis, and PeopleSoft or other book-length publication titles. In PeopleCode syntax, italic items are placeholders for arguments that your program must supply. We also use italics when we refer to words as words or letters as letters, as in the following: Enter the letter *O*.
KEY+KEY	Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press the W key.
`Monospace font`	Indicates a PeopleCode program or other code example.

Typographical Convention or Visual Cue	Description
“ ” (quotation marks)	Indicate chapter titles in cross-references and words that are used differently from their intended meanings.
. . . (ellipses)	Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax.
{ } (curly braces)	Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe (|).
[ ] (square brackets)	Indicate optional items in PeopleCode syntax.
& (ampersand)	When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object. Ampersands also precede all PeopleCode variables.

Visual Cues

PeopleBooks contain the following visual cues.

Notes

Notes indicate information that you should pay particular attention to as you work with the PeopleSoft system.

Note. Example of a note.

If the note is preceded by *Important!,* the note is crucial and includes information that concerns what you must do for the system to function properly.

Important! Example of an important note.

Warnings

Warnings indicate crucial configuration considerations. Pay close attention to warning messages.

Warning! Example of a warning.

Cross-References

PeopleBooks provide cross-references either under the heading “See Also” or on a separate line preceded by the word *See.* Cross-references lead to other documentation that is pertinent to the immediately preceding documentation.

Country, Region, and Industry Identifiers

Information that applies only to a specific country, region, or industry is preceded by a standard identifier in parentheses. This identifier typically appears at the beginning of a section heading, but it may also appear at the beginning of a note or other text.

Example of a country-specific heading: "(FRA) Hiring an Employee"

Example of a region-specific heading: "(Latin America) Setting Up Depreciation"

Country Identifiers

Countries are identified with the International Organization for Standardization (ISO) country code.

Region Identifiers

Regions are identified by the region name. The following region identifiers may appear in PeopleBooks:

- Asia Pacific
- Europe
- Latin America
- North America

Industry Identifiers

Industries are identified by the industry name or by an abbreviation for that industry. The following industry identifiers may appear in PeopleBooks:

- USF (U.S. Federal)
- E&G (Education and Government)

Currency Codes

Monetary amounts are identified by the ISO currency code.

Comments and Suggestions

Your comments are important to us. We encourage you to tell us what you like, or what you would like to see changed about PeopleBooks and other PeopleSoft reference and training materials. Please send your suggestions to:

PeopleSoft Product Documentation Manager PeopleSoft, Inc. 4460 Hacienda Drive Pleasanton, CA 94588

Or send email comments to doc@peoplesoft.com.

While we cannot guarantee to answer every email message, we will pay careful attention to your comments and suggestions.

Common Elements Used in PeopleBooks

As of Date	The last date for which a report or process includes data.
Business Unit	An ID that represents a high-level organization of business information. You can use a business unit to define regional or departmental units within a larger organization.
Description	Enter up to 30 characters of text.
Effective Date	The date on which a table row becomes effective; the date that an action begins. For example, to close out a ledger on June 30, the effective date for the ledger closing would be July 1. This date also determines when you can view and change the information. Pages or panels and batch processes that use the information use the current row.
Once, Always, and **Don't Run**	Select Once to run the request the next time the batch process runs. After the batch process runs, the process frequency is automatically set to Don't Run. Select Always to run the request every time the batch process runs. Select Don't Run to ignore the request when the batch process runs.
Process Monitor	Click to access the Process List page, where you can view the status of submitted process requests.
Report Manager	Click to access the Report List page, where you can view report content, check the status of a report, and see content detail messages (which show you a description of the report and the distribution list).
Request ID	An ID that represents a set of selection criteria for a report or process.
Run	Click to access the Process Scheduler request page, where you can specify the location where a process or job runs and the process output format.
SetID	An ID that represents a set of control table information, or TableSets. TableSets enable you to share control table information and processing options among business units. The goal is to minimize redundant data and system maintenance tasks. When you assign a setID to a record group in a business unit, you indicate that all of the tables in the record group are shared between that business unit and any other business unit that also assigns that setID to that record group. For example, you can define a group of common job codes that are shared between several business units. Each business unit that shares the job codes is assigned the same setID for that record group.
Short Description	Enter up to 15 characters of text.
User ID	An ID that represents the person who generates a transaction.

See Also

Enterprise PeopleTools 8.46 PeopleBook: PeopleSoft Process Scheduler

Enterprise PeopleTools 8.46 PeopleBook: Using PeopleSoft Applications

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 the PeopleSoft Customer Connection website:

Resource	Navigation
Hardware and software requirements	Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation and Software, Hardware and Software Requirements
Installation guides	Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation and Software, Installation Guides and Notes
PeopleBook documentation updates	Support, Documentation, Documentation Updates
Troubleshooting information	Support, Troubleshooting

CHAPTER 1

Getting Started with PeopleSoft MultiChannel Framework

This chapter discusses:

- PeopleSoft MultiChannel Framework overview.
- PeopleSoft MultiChannel Framework implementation.

PeopleSoft MultiChannel Framework Overview

PeopleSoft MultiChannel Framework provides the tools that are required to support multiple channels of communication between customers (users) and agents. Some PeopleSoft applications use PeopleSoft MultiChannel Framework, such as an email response management system (ERMS) from PeopleSoft CRM, or you can develop your own applications on the provided 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. The channels supported include email, web-based chat, voice, instant messaging, and generic events. PeopleSoft MultiChannel Framework can be used from any PeopleSoft application.

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 steps that are required to implement PeopleSoft MultiChannel Framework (MCF).

Depending on your business use of PeopleSoft MultiChannel Framework, 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 also develop your own applications built on the PeopleSoft MultiChannel Framework.
- If you are using PeopleSoft CTI, 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.

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

See Chapter 5, "Configuring REN Servers," page 49.

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 MCF Clusters," page 65 and Chapter 7, "Configuring MCF Queues," page 73.

Creating and Defining Agents

Agents are PeopleSoft users that 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 MCF Agents," page 89.

Developing for MCF

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 *Enterprise PeopleTools 8.46 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.

See Chapter 9, "Administering PeopleSoft MultiChannel Framework," page 99; Chapter 11, "Developing for PeopleSoft MultiChannel Framework," page 121; *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference*, "Mail Classes"; *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference*, "MCFIMInfo Class" and *Enterprise PeopleTools 8.46 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 CTI," page 11.

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

- PeopleSoft MultiChannel Console.

PeopleSoft MultiChannel Elements

PeopleSoft Multichannel Framework comprises the following services and elements:

- Universal queue, 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, 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 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:

- Computer-telephony integration (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 MultiChannel Framework.

Voice

The agent console offers a softphone and full CTI support with Genesys and Cisco CTI middleware, or any middleware implementing the PSMCAPI. Relevant application pages appear based on data attached to the call by the 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 conference peers and supervisors into a chat and transfer chat sessions to other agents or queues. Agents can also initiate collaborative chats with other agents on their buddy lists.

Email

PeopleSoft MultiChannel Framework enables applications to fetch Multipart Internet Mail Extensions emails from Post Office Protocol 3 and Internet Message Access Protocol 4 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 and role-based security before retrieving an attachment. The email framework is built on PeopleSoft Integration Broker technology.

PeopleSoft MultiChannel Framework also supports both inbound and outbound HTML email.

Generic Channel

Channels that are not provided by PeopleSoft MultiChannel Framework can be integrated by using the generic channel to enqueue tasks onto the universal queue.

Instant Messaging

PeopleSoft MultiChannel Framework includes the capability to use instant messaging called from an application page and using one of several instant messaging clients.

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.

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 accepting 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.

The CTI console can be customized using the JSMCAPI.

PeopleSoft MultiChannel Framework Architecture

This section discusses:

- PeopleSoft MultiChannel Framework server architecture.
- Chat architecture.
- Email architecture.

PeopleSoft MultiChannel Framework 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, as the browser windows maintain persistent connections to the REN server. Events can be sent proactively to browser windows without polling or page refreshes.

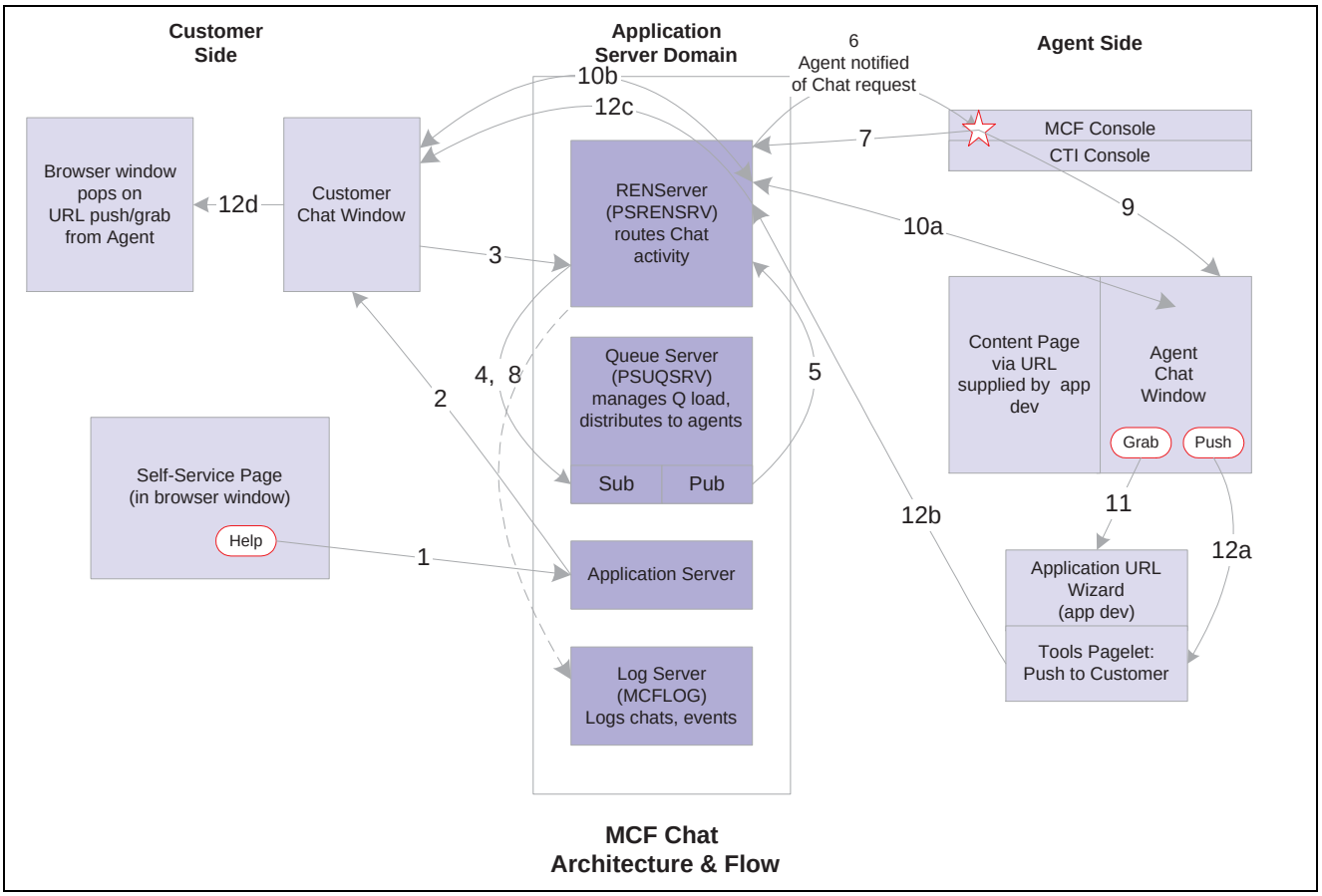
Applications send interaction and action requests (tasks) received from the supported communication channels to logical queues. The REN server notifies the universal queue server (the PSUQSRV process) responsible for that queue that a new task has arrived. Tasks are queued in order of priority until 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. Configure logging levels on the MCF administration pages.

Chat Architecture

The following diagram illustrates the architecture of a chat session:



Chat architecture and flow

1. When a customer clicks the Help button on an application page, InitChat() passes the following parameters to the application server:
   - Queue number.
   - Priority override.
   - Context page URL.
   - Query.

   See *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions".
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 there is a customer chat 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.

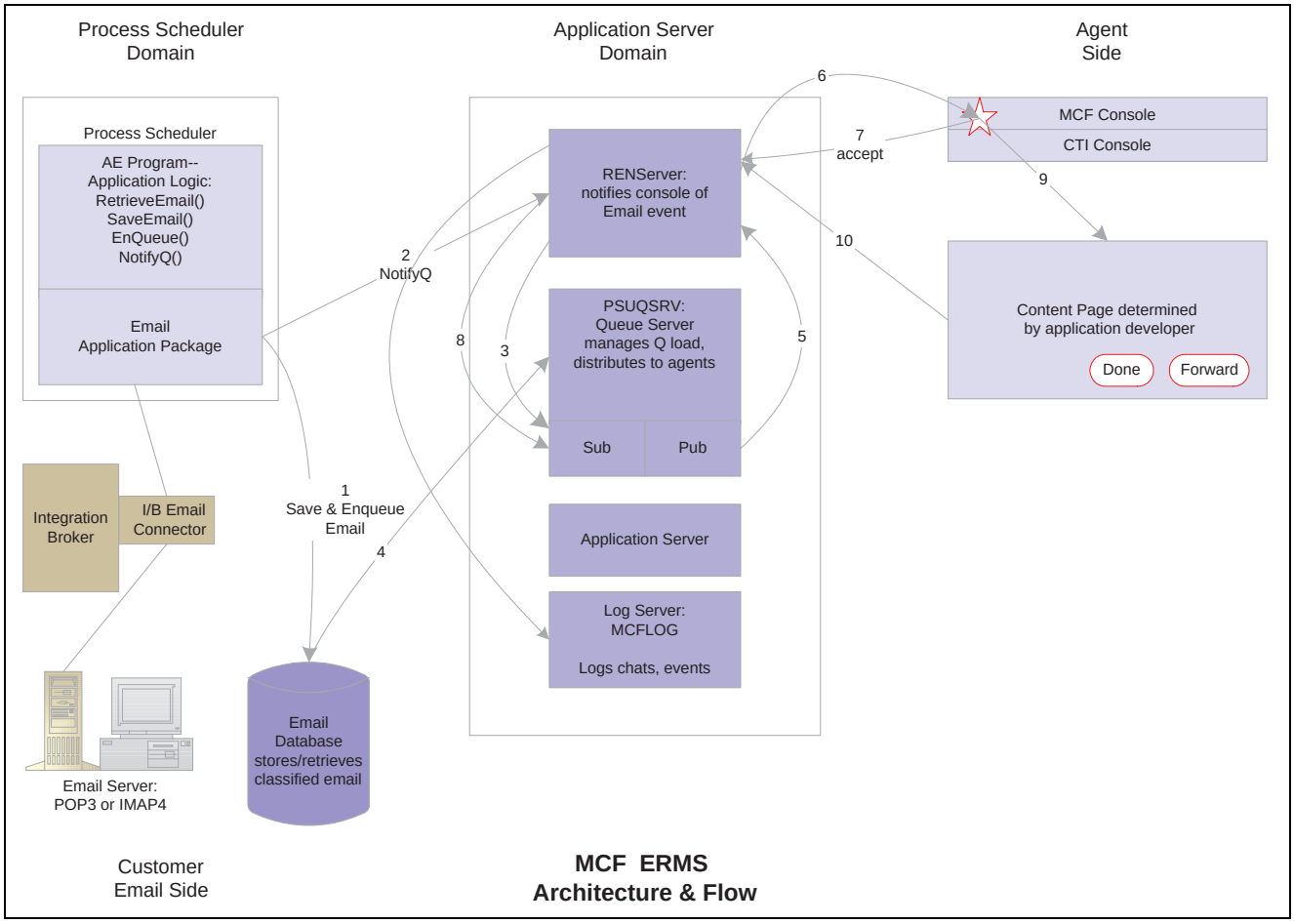
7. The agent, from the MultiChannel Console, notifies the REN server to notify the queue server that the agent has accepted the work.
8. The REN server notifies the queue that the agent has accepted the work.
9. The MultiChannel Console displays an agent chat window.

   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 following diagram illustrates the architecture of email processing:



Email architecture and flow

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.
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.

CHAPTER 3

Configuring PeopleSoft CTI

This chapter provides an overview of the PeopleSoft Computer Telephony Integration (CTI) console and PeopleSoft CTI components and discusses how to:

- Configure PeopleSoft CTI
- Configure queues and agents
- Use other CTI options
- Use the CTI sample page

Understanding PeopleSoft CTI

This section discusses:

- PeopleSoft CTI
- PeopleSoft CTI components.

PeopleSoft CTI

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

- Seamlessly integrates your PeopleSoft application with the Genesys CTI Framework, Cisco Intelligent Contact Management (ICM) software, or other CTI software that implements the PeopleSoft MultiChannel Application Programming Interface (PSMCAPI), 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.

PeopleSoft CTI is an optional component that you can integrate with the PeopleSoft MultiChannel Framework. That is, you have the option to incorporate a CTI channel within the MultiChannel Framework. PeopleSoft CTI requires third-party middleware in the form of the Genesys CTI Framework, a Cisco ICM system, or CTI middleware that implements PSMCAPI.

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.

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 their 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 there is an incoming call 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 using the CTI middleware.

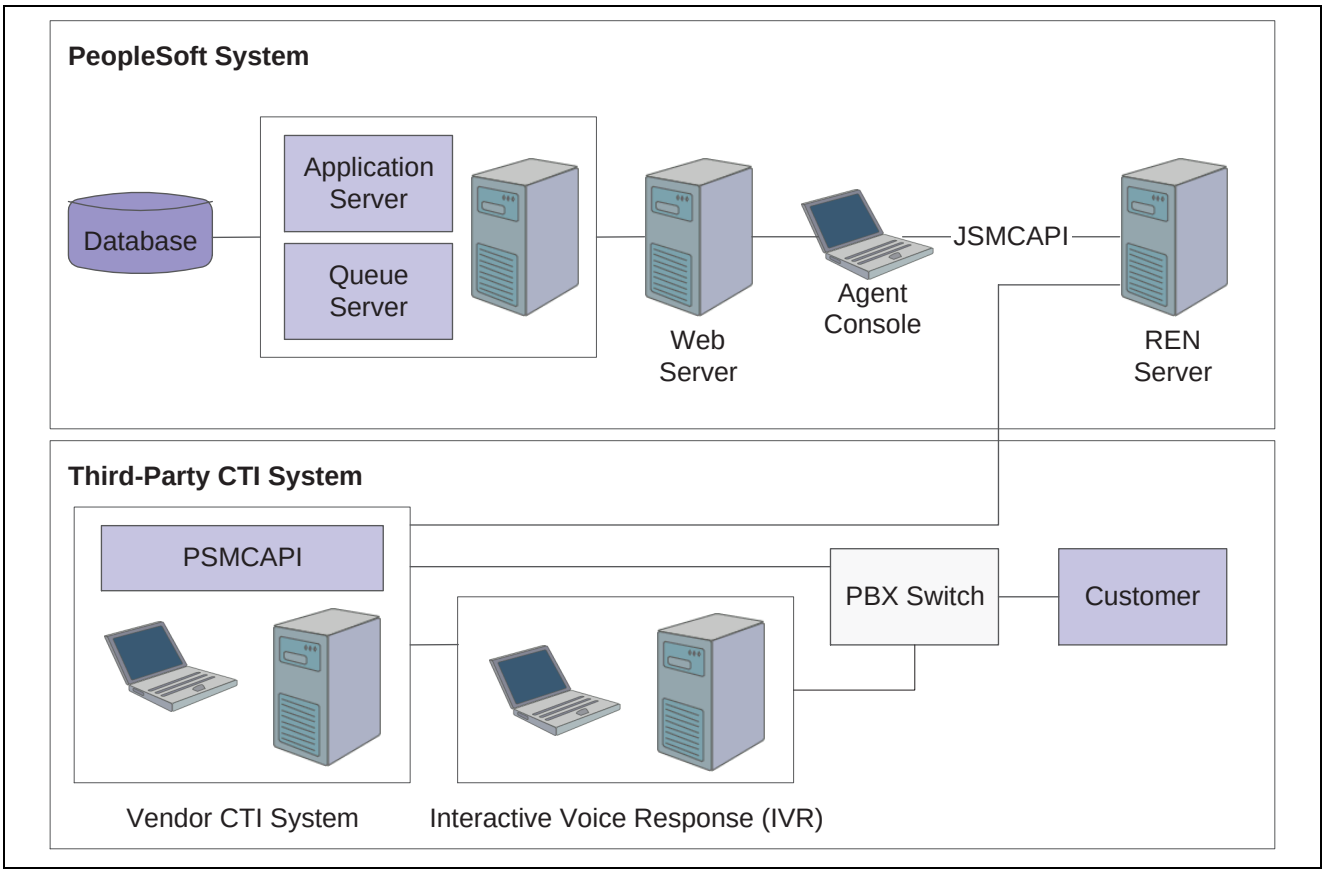
For PeopleSoft CTI systems using Enterprise PeopleTools 8.45.05 or later, the CTI console uses a JavaScript MultiChannel Application Programming Interface (JSMCAPI) to communicate with CTI middleware that implements the PeopleSoft MultiChannel Application Programming Interface (PSMCAPI).

For PeopleSoft CTI systems using Enterprise PeopleTools 8.44 or earlier, 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 that is approximately 500KB. The applet resides in the browser cache to reduce network traffic and improve response time.

Note. The CTI console Java applet is deprecated in PeopleTools 8.46.

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

This diagram illustrates PeopleSoft CTI architecture using PSMCAPI and JSMCAPI:



PeopleSoft CTI integration architecture

Note. PeopleSoft supports both Cisco and Genesys CTI solutions using the CTI Java applet, and any CTI solution that implements the PSMCAPI. In this document, when the phrase "your CTI vendor" or "CTI middleware" appears, assume it refers to either Cisco or Genesys products, or other CTI vendors supporting the PSMCAPI, as appropriate to your installation.

See Also

Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference, "Internet Script Classes (iScript)"

Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference, "Internet Script Classes (iScript)"

Understanding PeopleSoft CTI Components

This section describes components required for PeopleSoft CTI.

This section discusses:

- PeopleSoft requirements.
- PeopleSoft MultiChannel API.
- JavaScript MultiChannel API.
- Required security for PSMCAPI.

- Genesys components.
- Cisco components.
- Java Runtime Environment.

PeopleSoft CTI Requirements

PeopleSoft CTI requires either:

- CTI middleware implementing the PSMCAPI, or
- Genesys CTI and the Java Runtime Environment, or
- Cisco CTI and the Java Runtime Environment.

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.

No additional installation is required to use PSMCAPI or JSMCAPI; they are installed during PeopleTools installation.

The following table provides the locations of PSMCAPI and JSMCAPI components:

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

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.

Required Security for PSMCAPI

PeopleSoft CTI solutions that use the PeopleSoft MultiChannel Application Programming Interface (PSMCAPI) require CTI-specific configuration of REN server security to authenticate the CTI server. The CTI server is not part of a PeopleSoft application server domain, but it can generate an authentication token (PSTOKEN) to be authenticated by the application server. To enable this authentication, PeopleSoft delivers a default CTI server permission list and role for use exclusively by the CTI server.

You must create a CTI server user profile (such as CTISRV) and assign the CTI server role, PeopleTools CTI Server, to it. Be sure to specify the user profile password and symbolic ID, and set the ID type as *None.*

The following table lists CTI server security objects for use with PSMCAPI:

Security Object	Name
CTI server permission list	PTPT1700
CTI server security object (set for full access on PeopleTools, Realtime Event Notification Permissions page for specified permission list)	MCF CTI Server
CTI server role	PeopleTools CTI Server

CTI agents only require 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," page 49

Enterprise PeopleTools 8.46 PeopleBook: Security Administration

JavaScript MultiChannel API

The JavaScript MultiChannel Application Programming Interface (JSMCAPI) is an interface that can be used by application developers 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.

The JSMCAPI CTI console replaces the CTI Java applet, which will be deprecated in PeopleTools 8.46.

See Also

Chapter 11, "Developing for PeopleSoft MultiChannel Framework," Using and Demonstrating JSMCAPI, page 129

Genesys Components

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

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 PeopleSoft supports, refer to the PeopleSoft Platforms database on the Customer Connection website.

Note. The Genesys Configuration Server is an optional but supported component.

Cisco Components

If you are using a Cisco CTI system with the PeopleSoft CTI Java applet console, PeopleSoft assumes that you already have a functioning and appropriately licensed Cisco system configured at your site. PeopleSoft ships the Cisco Java API as part of the CTI applet, however, PeopleSoft ships no other Cisco products.

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 PeopleSoft supports, refer to the PeopleSoft Platforms database on the Customer Connection website.

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 Java Runtime Environment is required for those PeopleSoft CTI systems using the Java applet console. The JRE is not required if you are using the JSMCAPI console.

PeopleSoft recommends that you install and use the Sun Java Plug-in 1.4.2 or later, which you can download from www.java.com.

Applet Protection

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

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 code signed by PeopleSoft for all subsequent sessions or just for the current session only. Instruct your end users to select the appropriate option for your site.

See Also

Supported web browsers on the Customer Connection website.

Browser limitations on the Customer Connection website.

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.

To configure PeopleSoft CTI, use the CTI Configuration (PT_CTI_CONFIG) and Miscellaneous (PT_CTI_MISC)components.

This section discusses how to:

- Install PeopleSoft CTI.
- Enable PeopleSoft CTI.
- Create CTI configurations.
- Configure the CTI non-applet console.
- Configure the CTI applet console.
- Use the Shared Phone Book page.
- Use the CTI Genesys page.
- Use the CTI Cisco page.
- Use the Miscellaneous page.

Pages Used to Configure CTI

Page Name	Object Name	Navigation	Usage
CTI Type	CTICONSOLETYPE	PeopleTools, MultiChannel Framework, CTI, CTI Type	Specify whether you are using the CTI Java applet console or the CTI JavaScript (non-applet) console.
CTI Non-Applet	CTIREN	PeopleTools, MultiChannel Framework, CTI, CTI Non-Applet	Configure CTI for use with the JavaScript console.
CTI Applet	PT_CTI_CONFIG	PeopleTools, MultiChannel Framework, CTI, CTI Applet	Configure CTI for use with the Java applet console.
Shared Phone Book	PT_CTI_PNCTI	PeopleTools, MultiChannel Framework, CTI, Shared Phone Book	Create a list of frequently dialed phone numbers for use with a specified configuration.
CTI Cisco	PT_CTI_CISCO	PeopleTools, MultiChannel Framework, CTI, CTI Cisco	Configure parameters specific to Cisco CTI middleware for use with the Java applet console.
CTI Genesys	PT_CTI_GENESYS	PeopleTools, MultiChannel Framework, CTI, CTI Genesys	Configure parameters specific to Genesys CTI middleware for use with the Java applet console.
Miscellaneous	PT_CTI_MISC	PeopleTools, MultiChannel Framework, CTI Configuration, Miscellaneous	Specify default screen pop-up window URL.

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.

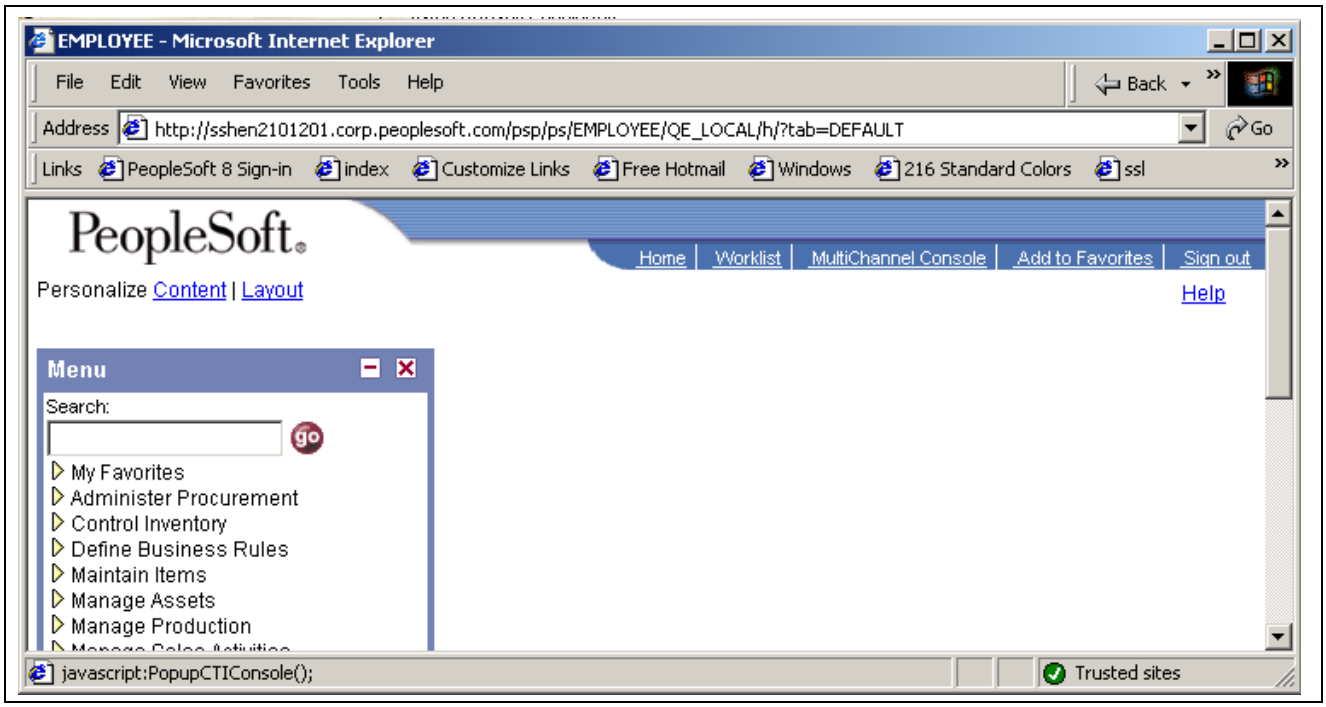
See Also

PeopleTools Installation Guide for your platform.

CTI vendor product documentation.

Enabling PeopleSoft CTI

If a user is set up as a CTI agent or a MultiChannel Framework agent, the user will see the following when signing on to PeopleSoft:



MultiChannel Console link

Notice the MultiChannel Console link in the universal navigation header. 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 log on to queues supported by the same REN server.

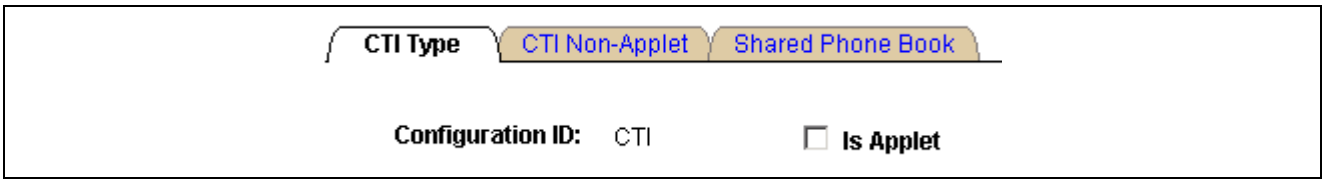
See Also

Enterprise PeopleTools 8.46 PeopleBook: Internet Technology, "Understanding Portal Technology"

Chapter 3, "Configuring PeopleSoft CTI," Configuring CTI Agents, page 28

Creating CTI Configurations

Access the CTI Type page.



CTI Type page

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

CTI Type page

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

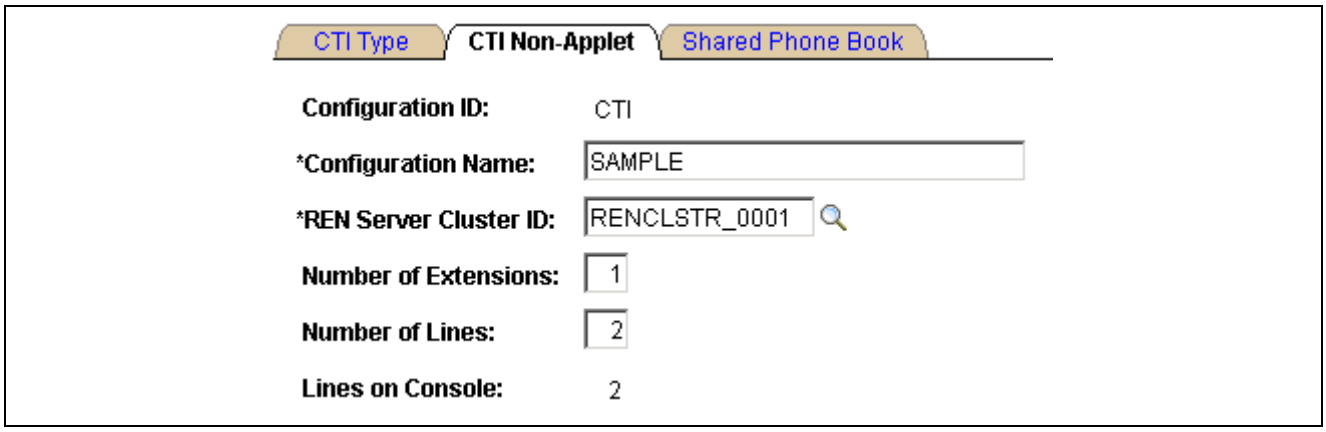
A CTI configuration contains all the information required for a user to be able to connect to one of the following:

- Any CTI server that implements PSMCAPI.
- Genesys T-Server or Configuration Server.
- Cisco ICM Server.

To create a CTI configuration, use the CTI Type page, the CTI Non-Applet or CTI Applet page, the Shared Phone Book page. If you are using the CTI applet, also configure the CTI Cisco or CTI Genesys page, depending on your CTI middleware.

Configuring the CTI Non-Applet Console

Access the CTI Non-Applet page to configure the CTI non-applet (JSMCAPI) console.



CTI Non-Applet page

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

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

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

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

For Cisco, only a value of *1* can be used.

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

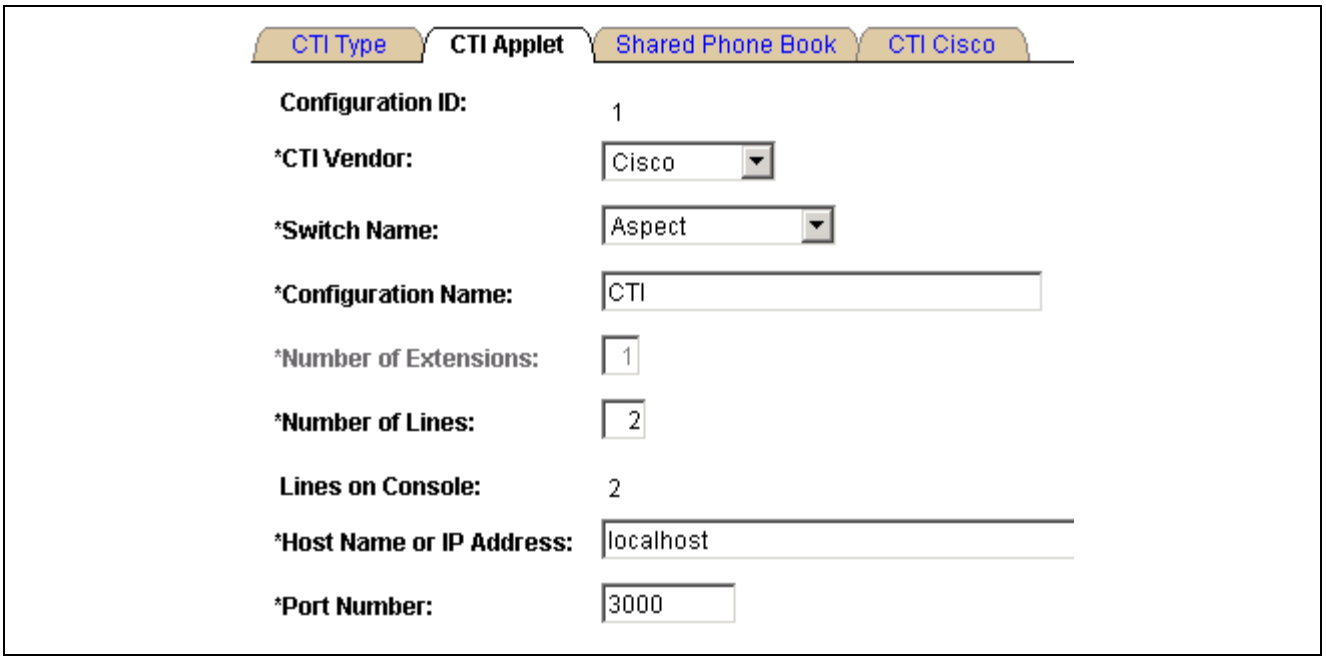
If Number of Extensions is *1*, this value should be *2*.

If Number of Extensions is *2*, this value should be *1*.

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

Configuring the CTI Applet Console

Access the CTI Applet page to configure PeopleSoft CTI for use with the CTI applet console.



CTI Applet page

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

Currently, the supported switches for Genesys are:

- *Siemens Hicom 300*
- *Nortel Meridian*
- *Avaya Definity G3*
- *Aspect*

Consult the Genesys Framework T-Server Reference Manuals to check if features offered by the CTI Console are supported on your switch. For example, mute transfers are not supported on Aspect switches.

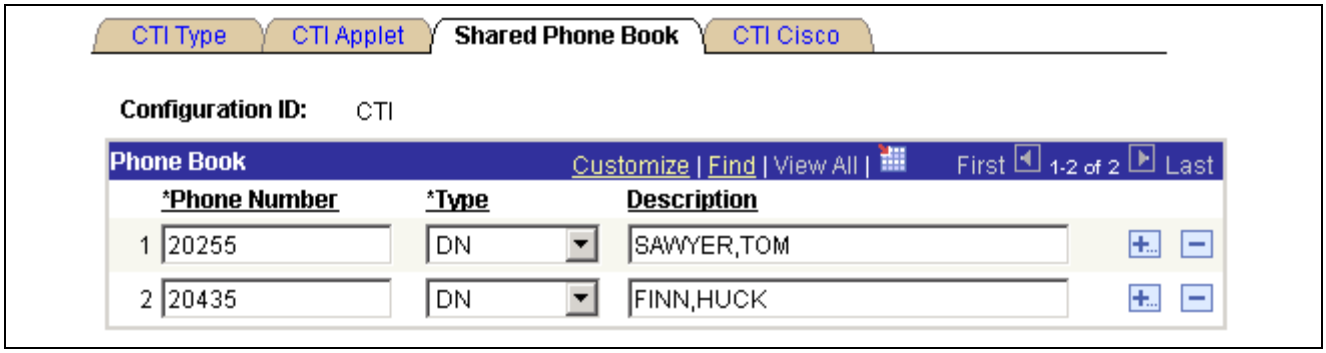
Currently, the supported switches for Cisco are:

- *IPCC*
- *Avaya Definity G3*
- *Aspect*

	Consult the Cisco documentation to determine if features offered by the CTI Console are supported on your switch. For example, DND and mute transfers may not be supported for all switches.
Configuration Name	Add a descriptive name to help identify the configuration.
Number of Extensions	Number of extensions or directory numbers associated with the telephone. For Genesys, *1* or *2* can be used. For Cisco, only a value of *1* can be used.
Number of Lines	Number of lines associated with each extension. Depending on your configuration, you can specify up to two lines. If Number of Extensions is *1*, this value should be *2*. If Number of Extensions is *2*, this value should be *1*.
Lines on Console	The CTI console supports up to two lines. The only supported configurations are two extensions with one line each, or one extension with two lines, or one extension with one line.
Host Name or IP Address	Enter the host name or IP address for one of the following: • Genesys T-Server or Configuration Server. • Cisco ICM Server.
Port Number	Enter the port number on which the T-Server or Configuration Server listens.

Using the Shared Phone Book Page

Access the Shared Phone Book page.



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 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.

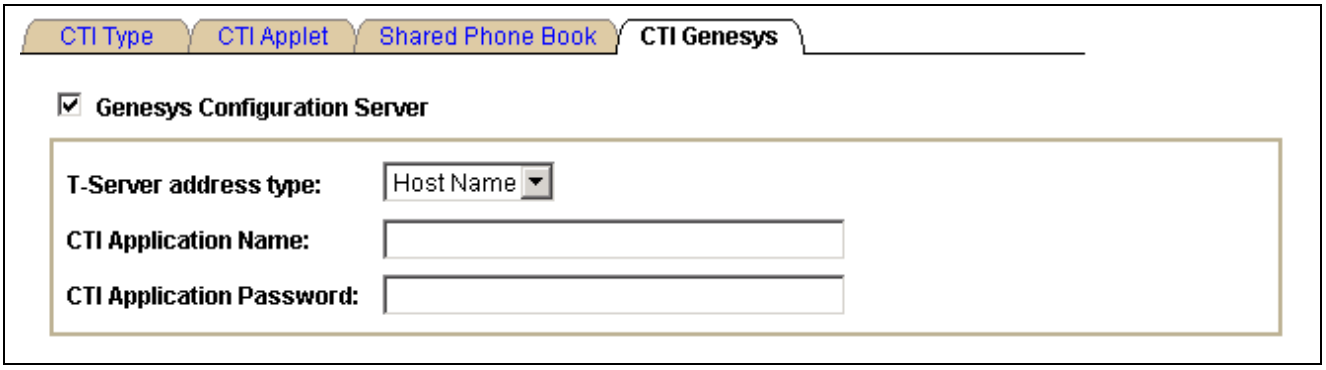
Phone Number	Enter a frequently dialed phone number to associate with this configuration.
Type	Select either:

- 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 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.

CTI Genesys Page

Access the CTI Genesys page.



CTI Genesys page

The CTI Genesys page provides additional options for Genesys implementations. This page only appears when Genesys is selected as the CTI Vendor on the General Configuration page.

Genesys Configuration Server Directs 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 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.

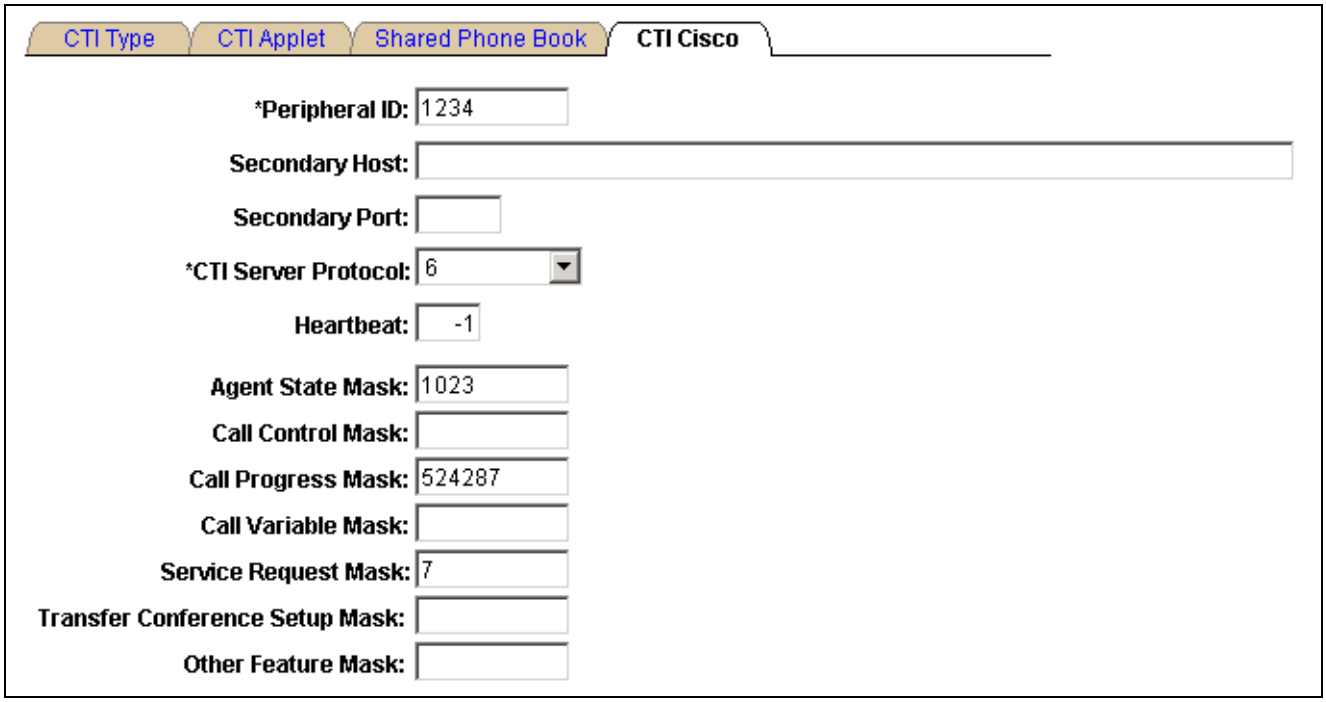
T-Server address type If you are using a configuration server to indicate what T-Server to connect to (instead of connecting directly to a T-Server), indicate whether the configuration server is returning the host name or the IP address of the T-server.

Specify the type of address that should be used 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 on to the Genesys Configuration Server. Using this option, each user connects to the configuration server with the same application name.

CTI Cisco Page

Access the CTI Cisco page.



CTI Cisco page

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.

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

Secondary Host and **Secondary Port** Specify 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 Specify an interval in seconds for the system to send a message to the CTI Server to make sure it is up and 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 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. PeopleSoft 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 may be overridden by a Cisco system-level mask.

Refer to your Cisco documentation for more information.

Call Control Mask

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

Refer to your Cisco documentation for information on this mask.

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

Call Progress Mask

Specifies any unsolicited call event messages that you want your application to receive. For instance, you can specify 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.

PeopleSoft 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 may 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 on this mask.

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

Service Request Mask Enables you to adjust CTI service masks. PeopleSoft 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 on this option.

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

Transfer Conference Setup Mask Specifies the valid ways the application may be configured for a transfer or conference call.

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

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

Refer to your Cisco documentation for more information on 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.

Miscellaneous Page

Access the Miscellaneous page.

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

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

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

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

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

Note. To ensure that the user does not have to sign on to the pop-up window, the domain of the default URL must exactly match the domain of the signon URL. If there is no domain for the signon URL, the machine name of the default URL should be the same as the machine name of the signon 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

Enterprise PeopleTools 8.46 PeopleBook: Security Administration, "Setting up Digital Certificates and Single Signon"

Configuring Queues and 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.
- Use the Personalization page.

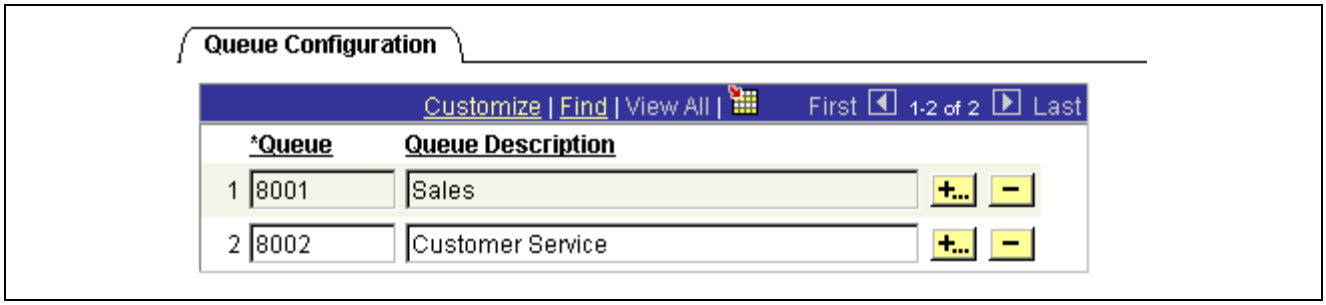
- View information on the Agent Info page.

Pages Used to Configure Queues and Agents

Page Name	Object 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 agent ID and queue information.
Phone Book	PT_CTI_PNAGENT	PeopleTools, MultiChannel Framework, CTI Configuration, Agent, Phone Book	Define an agent phone book.
Personalization	PT_CTI_AGENTGUI	PeopleTools, MultiChannel Framework, CTI Configuration, Agent, Personalization	Specify the location of 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.



Queue Configuration page

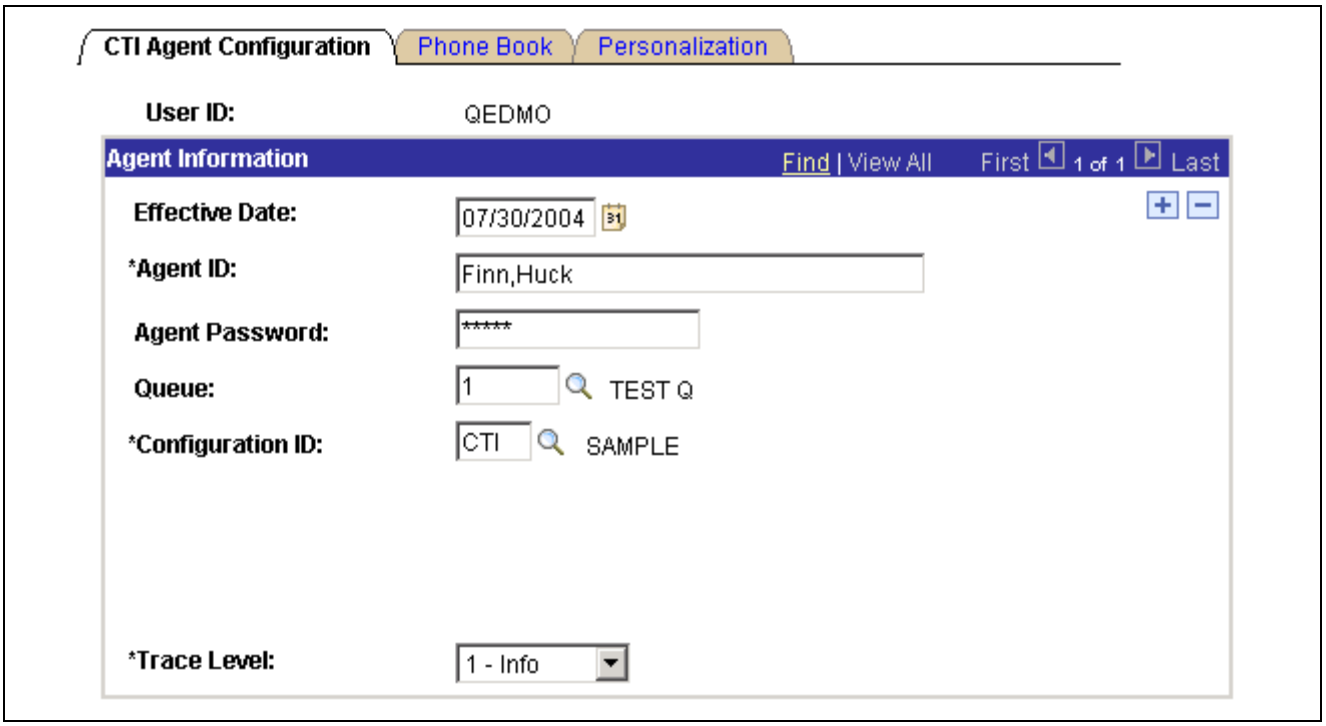
Use this page to add queues for agents.

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

Queue Description Specify a brief description of the queue.

Configuring CTI Agents

Access the Agent page.



CTI Agent Configuration page

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 Specify a user ID for the agent within the switch.

Agent Password Enter the password the agent uses to sign on to the phone, if any.

Queue Enter the name of the queue 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 you created using the CTI Applet or CTI Non-Applet page.

Application User Name/Password Enter the CTI user name and password of the agent.

CTI Client Signature

Note. This option only appears 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.

Trace Level For Trace Level you have the following options:

- 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 regarding a CTI issue, include a level 2 - Debug trace.

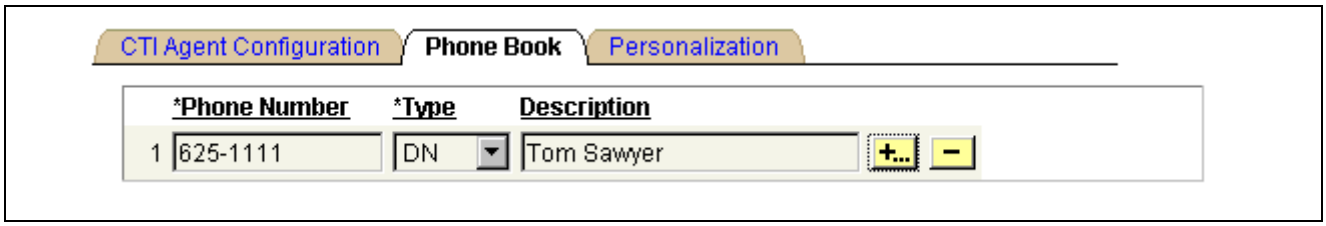
If you are using the JSMCAPI (non-applet) console, two tracers (a JSMCAPI tracer and a console tracer) appear if trace level 2, Debug, is selected.

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.

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

Using the Phone Book Page

Access the Phone Book page.



Phone Book page

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 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.

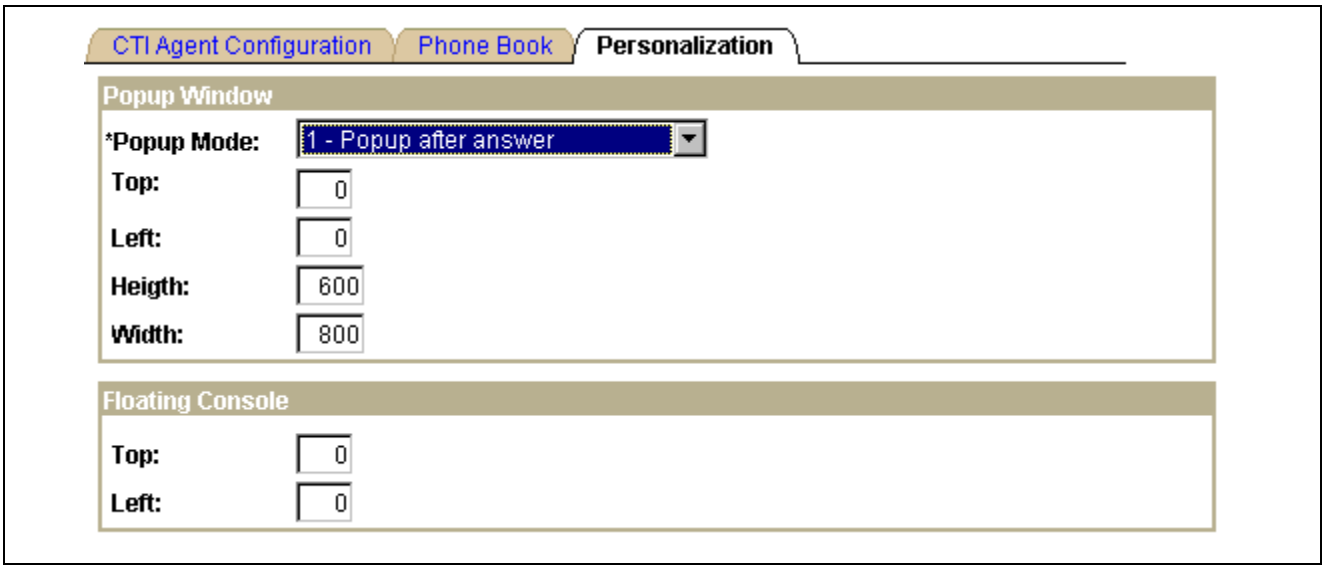
Type Select either:

- 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.

Using the Personalization Page

Access the Personalization page.



Personalization page

Use this page to personalize the pop-up windows' timing, size, and position 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 there is an incoming call. If it appears before the call is answered the agent can determine if they should answer the call based on the information displayed 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 screen pop-up 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 screen pop-up.

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

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

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

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

Logout when console is closed Select to automatically log the agent out when the CTI console is closed. Only applies on a Genesys switch system; Cisco always logs out the CTI agent when the CTI console is closed.

Floating Console

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

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

Viewing Information on the Agent Information Page

Access the Agent Information page.

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

User ID The agent's PeopleSoft User ID.

Agent ID 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 CTI Options

You can configure other optional CTI parameters.

This section discusses how to:

- Configure pop-up windows.
- Support single signon.
- 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 it 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-ups from your site or domain.

Setting Up Genesys for Pop-up Windows

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 displays for the agent. This data is typically created and attached to the call using the Genesys Strategy Builder.

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 menu within PeopleSoft 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 executed through the screen pop-up.

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

Unless you choose to use the default URL for your pop-ups windows, you must create certain expanded call context (ECC) variables. Expanded call context (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 PeopleSoft can receive the attached call data from the IVR.

PeopleSoft only recognizes 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 chars. The valid values are: Panel (page) or Script. It represents the type of PeopleSoft service being called, either a panel (page) or a script.

If ICType is set to panel, the following variables apply:

- user.PS.Component. 20 chars. This variable is required to specify the name of the destination component.
- user.PS.Menu. 10 chars. This variable is required to specify the name of the menu within PeopleSoft containing the destination component.
- user.PS.Market. 3 chars. This variable is required to specify the name of the Market of the destination component.

If the ICType is set to Script, then the following variable applies:

user.PS.ICScriptProgramName. 70 chars. If the ICTYPE is set to Script, then this variable is required to specify the location of the iScript, which is executed through the screen pop-up.

Descr

user.PS.Descr. 30 chars.

You can add any descriptive information that might be useful to the agent. This information displays 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 the *user.PS.* before passing the parameter to the application. For example,

user.PS.App1(3 chars)

user.PS.App2(1 chars)

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

Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference, "Internet Script Classes (iScript)"

Supporting Single Signon

PeopleSoft CTI offers single signon. The console connects to the CTI middleware using the CTI user ID and password retrieved from the PeopleSoft database.

See Also

Enterprise PeopleTools 8.46 PeopleBook: Security Administration, "Setting up Digital Certificates and Single Signon"

CTI Event Logging

CTI events can be logged in the CTI Event Log. The logged CTI events can be viewed 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.

See Also

Chapter 7, "Configuring MCF Queues," Tuning Cluster Parameters, page 81

Implementing "Free Seating"

When users sign on, 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.

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

Using the CTI Sample Pages

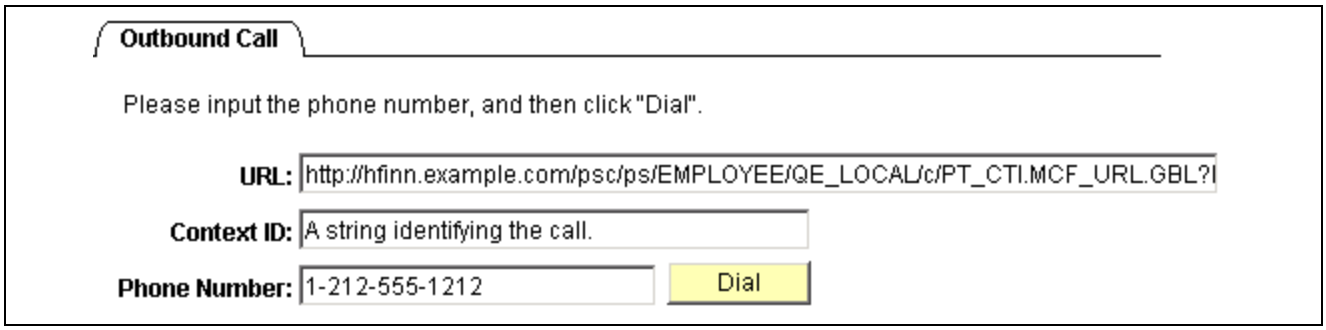
To demonstrate an outbound call, use the Sample Pages component (PT_CTI_DEMOOUTB). The CTI sample pages are intended for demonstration purposes only and should not be used in production.

Page Used to Demonstrate Outbound Calls

Page Name	Object 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.



Outbound Call page

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 Specify the URL of a page to display when dialing out.

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

Phone Number Enter the telephone number 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 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.

See Also

Chapter 11, “Developing for PeopleSoft MultiChannel Framework,” Using the CTI Sample Console, page 131

CHAPTER 4

Using PeopleSoft CTI

This chapter provides an overview of PeopleSoft CTI and discusses how to use PeopleSoft CTI.

Understanding PeopleSoft CTI

This section discusses:

- PeopleSoft CTI.
- The CTI interface.

PeopleSoft CTI

PeopleSoft CTI is a browser-based call management system that helps call agents work more efficiently with customers. PeopleSoft CTI integrates your Genesys or Cisco CTI middleware 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.

Note. PeopleSoft CTI supports both Genesys and Cisco CTI systems, and any CTI system that implements the PeopleSoft MultiChannel Application Programming Interface (PSMCAPI). In the following sections, when the phrase “your CTI middleware” or “CTI vendor” appears, assume that it refers to either Genesys or Cisco products, or another product implementing PSMCAPI, as appropriate for your installation.

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 behave differently.

The CTI Interface

The PeopleSoft CTI interface is incorporated *within* the MultiChannel Console. This document assumes you have read the documentation pertaining to the MultiChannel Console.

See Also

Chapter 3, "Configuring PeopleSoft CTI," page 11

Chapter 10, "Managing Tasks and Using Chat in PeopleSoft MultiChannel Framework," page 113

Cisco Switch Considerations

The following list presents the limitations of the Cisco Systems IP Contact Center switch:

- MAKE_CALL is only supported 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 as 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.
- Use the CTI console
- Select call actions.
- Use Cisco switches.
- Answer a call.
- Transfer a call.
- Initiate conference calls.
- Work with the hold status.
- Disconnect a caller.
- Switch agent ready status.
- Dial an outbound call.
- Complete a call.

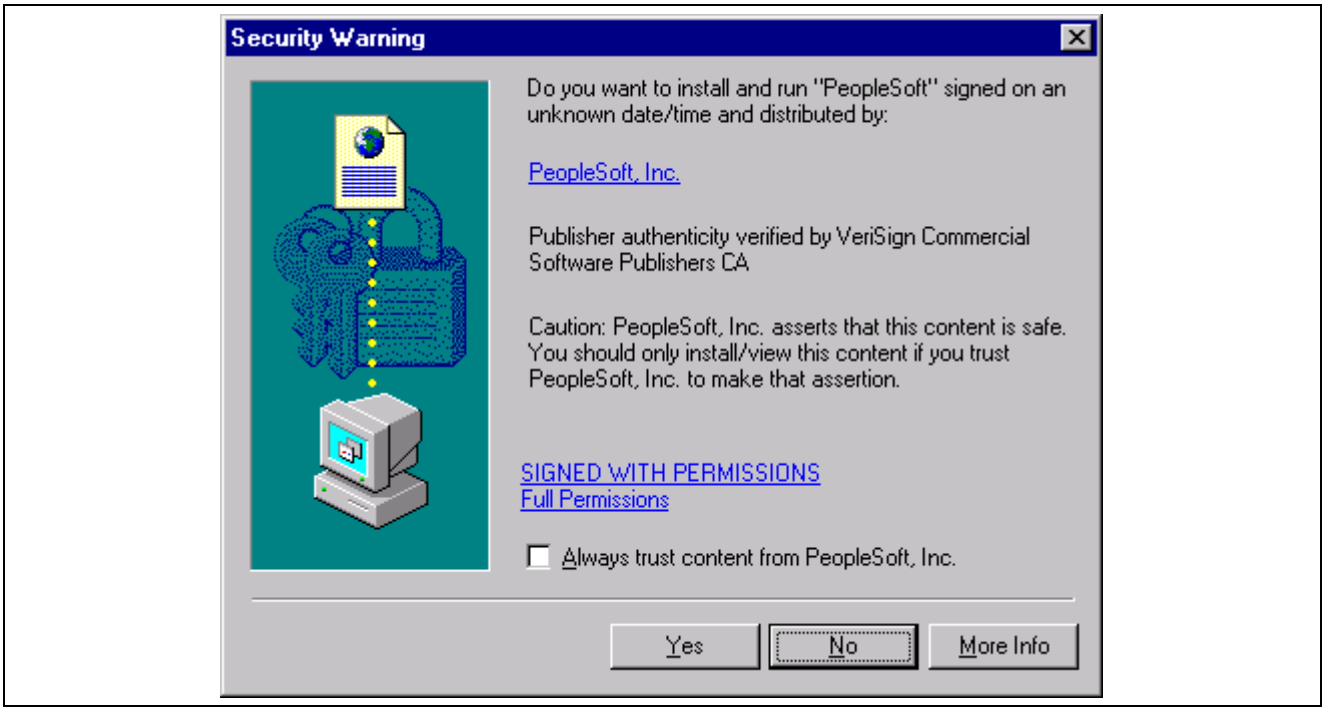
- Use hot keys.

Getting Started

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

- Signing in to PeopleSoft.
- Specifying your extension information.
- Connecting to your CTI middleware (either Genesys of Cisco).

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



Security Warning dialog box

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 PeopleSoft.
2. Click the MultiChannel Console link in the universal navigation header.
3. On the CTI control bar, set your configuration by doing the following:
   - Make sure that your agent ID appears beneath the CTI Agent ID label.

- Make sure 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, specify 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 Genesys or Cisco.

  The CTI console appears.

Using the CTI Console

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

Note. Keep in mind that the PeopleSoft CTI interface operates *within* the MultiChannel Framework Console. CTI users will only see the MultiChannel Console 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.

CTI Console

(register)

Enables you to register and deregister 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.

Lines

Select the option to the left of the telephone icon to activate the associated line. The icon you select determines the active line. All call actions 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 only valid 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. A CTI agent on a Genesys switch system can select on the agent Personalization page whether to log out automatically when the console is closed. Cisco always logs out the CTI agent when the CTI console is closed.

Selecting Call Actions

The Select Call Action drop-down list box contains all of the options 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 there are two buttons, Go and Cancel, that appear to the right of the Select Call Action drop-down list box. Go executes the selected call action, and Cancel stops the call action you executed.

The following list describes the call actions:

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

Answer Call Answers an incoming call. The Answer Call button displays and flashes only when there is an incoming call that 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. This action is not supported on Aspect switches.

Transfer Transfers the caller to the desired number. You will 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. Displays only when the caller is on hold.

Release Disconnects the caller. This option is only available as a selection once 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 keep in mind that they do not get the pop-up window showing customer data.

The following call actions are available depending on whether you are using a DN or Queue configuration:

Queue	*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.

Answering a Call

After you have logged on to PeopleSoft CTI, you are ready to 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 option 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 there is an incoming call.

2. Click Go.

   The popup 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.

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

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 initiating a transfer or conference call on the Cisco system and the contacted party does not answer, wait until the system notifies you of the unsuccessful connection before attempting another action. This can require 20 to 30 seconds.

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. Make sure the appropriate telephone line is selected and 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 for you when you first received the call.

To perform a transfer (consultative):

1. Make sure the appropriate telephone line is selected and 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 using 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. Once the outbound call is established you can consult with the recipient and place that call on hold. To complete the transfer, you need to go back to the first line and select *Complete* and click Go. This will release the call on the second line and transfer the call on the first line to the recipient of the transfer.

   When using two extensions each with one line, the CTI Console does not have access to the outbound call to the intended recipient of the transfer. Once the outbound call is established, you can consult with the recipient. You do not have to toggle between the two lines, and do not have the ability to put the recipient on hold.
5. To complete the transfer, select *Complete* and click Go.

Initiating Conference Calls

If you find that 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. Make sure that the appropriate telephone line is selected and 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 get 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 to 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 popup 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 it doesn't matter whether the mode is 0 or 1. In that case, the second agent gets the screen pop-up only after the first agent completes the transfer or conference call.

     When using one extension with two lines, the outbound call 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 go back to the first line and select *Complete* and click Go. This releases the call on the second line and starts the conference on the first line.

     When using two extensions each with one line, the CTI Console does not have access to the outbound call to the third party. Once the outbound call is established you consult with the target agent. You do not have to toggle between the two lines, and do not have the ability to 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.

Working with the Hold Status

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

To place a call on hold:

1. Make sure that the appropriate telephone line is selected and green.
2. From the Select Call Action drop-down list box select *Hold.*
3. Click Go.

To retrieve a call on hold:

1. Make sure that the appropriate telephone line is selected and yellow.

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

2. Click Go.

Disconnecting a Caller

After you have finished with a call, follow this procedure to release the call.

To release a call:

1. Make sure that you no longer require the call to remain active.
2. On the console control bar, select *Release* from the Lines drop-down list box.
3. Click Go.

   The system automatically places you in wrap-up mode, which enables you to complete any remaining work before accepting more incoming calls. Technically, when in wrap-up mode, your status is Agent Not Ready. 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.

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 will 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.

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. Make sure that you are in Agent Ready or Agent Not Ready status.
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, 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.
5. Click Go.

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

   Note. The system places you in Agent Not Ready status until you release the call.

   When making an outbound call:

   - A URL can be specified to display a page.

     When displaying the page, the outbound call data is also attached to the URL so that an application can collect information such as the ANI, 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 Cisco or Genesys 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 CTI," Using the Shared Phone Book Page, page 22

Completing a Call

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

When you become available to accept incoming calls depends on how your system administrators have set up your CTI middleware. Genesys has a setting known as *wrap-up time*, which allows an agent a certain amount of time to update the previous caller's information or complete related work before accepting more calls.

Using Hot Keys

Hot keys are combinations of keyboard buttons 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.
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 execute a call action.
ALT + C	Enables you to cancel a call action.
ALT + Z	Enables you to check agent status.

CHAPTER 5

Configuring REN Servers

This chapter provides an overview of real-time event notification (REN) servers, provides an example of configuring a reverse proxy server and REN servers, and discusses how to:

- Configure REN server security.
- Configure REN servers.
- Configure REN server clusters.

Understanding REN Servers

This section discusses:

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

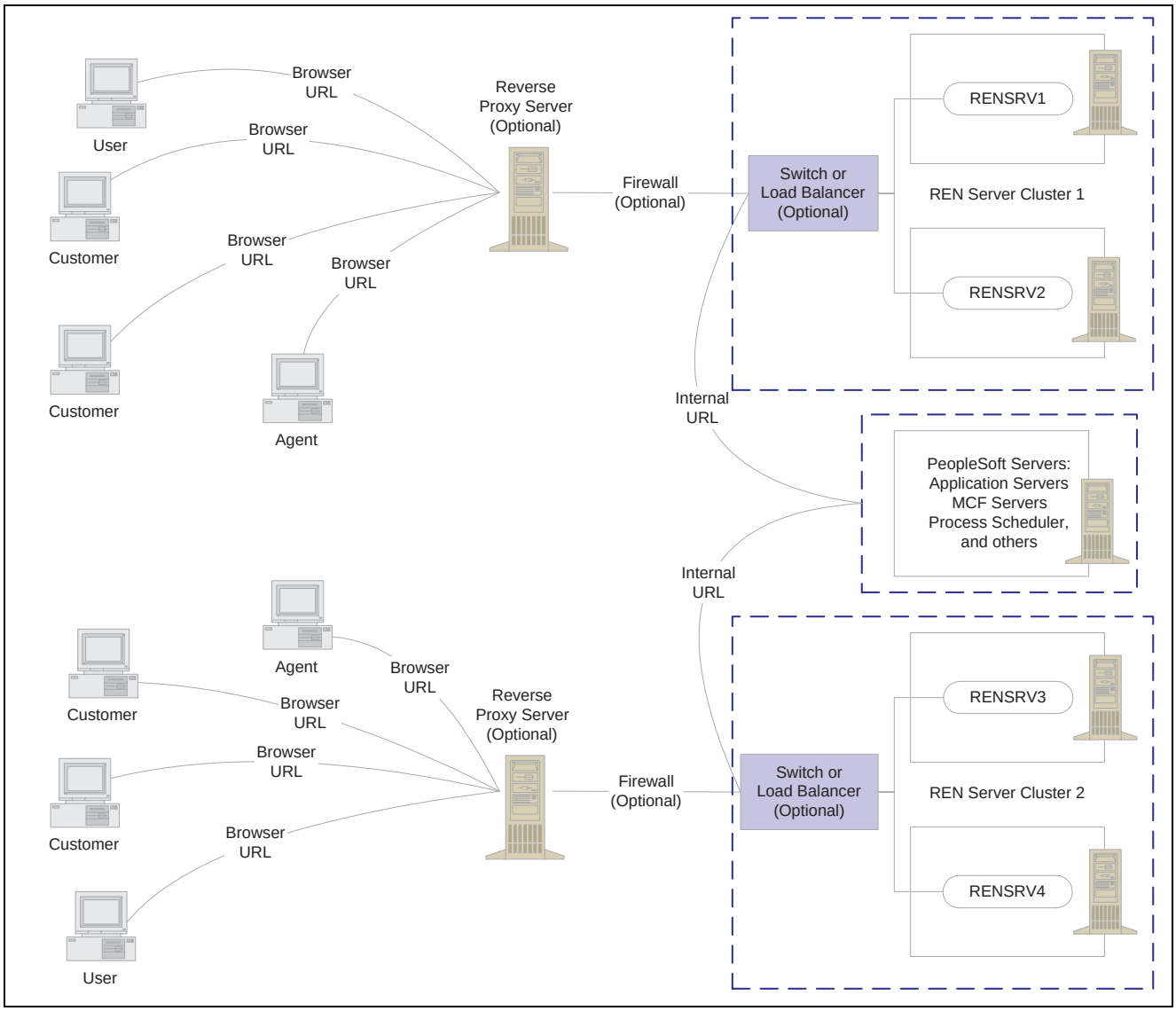
The REN server, an application server domain process, is essential to PeopleSoft MultiChannel Framework (MCF) architecture. MCF events are sent to REN servers, which deliver them to recipients of those topics.

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, as 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:



REN server configuration example

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. There are two scenarios for failure recovery:

- For a standalone REN server, BEA 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) will be 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 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.

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 in order 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 Secure Socket Layer (SSL), then the browser URL is an HTTPS address to a reverse proxy server or SSL accelerator.

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

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 using 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 currently logged into PeopleSoft with appropriate REN server permissions. You must enable single signon 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 signon is enabled.

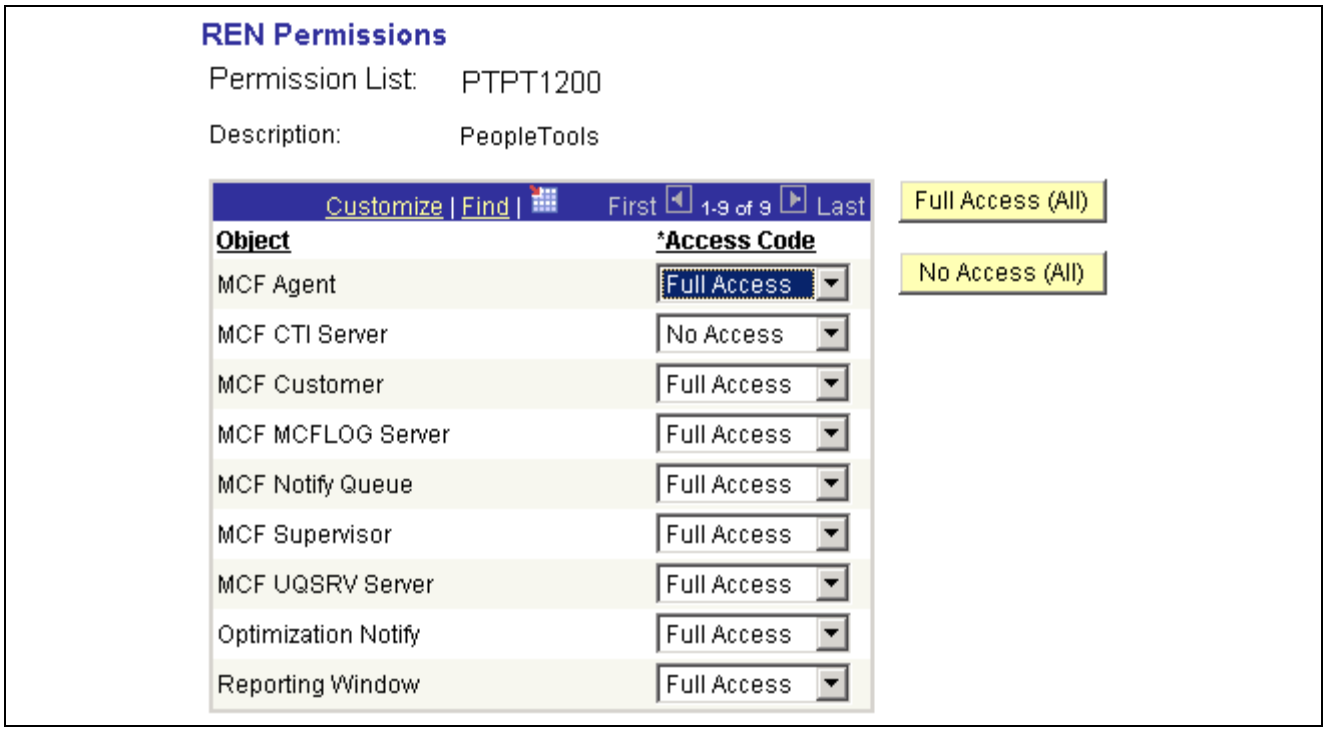
See Also

Enterprise PeopleTools 8.46 PeopleBook: Security Administration

Enterprise PeopleTools 8.46 PeopleBook: System and Server Administration

Defining Permission Lists for REN Server Access

Access the REN Permissions page.



REN Permissions page

Define access to REN servers in the appropriate permission lists.

To define access to REN servers in a permission list:

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 defined with the current permission list, select Full Access for each object 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.

Note. To enable access to the Report-to-Window functionality, add WEBLIB_RPT to the permission list's Web Libraries page, 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 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 will stay in a status of queued.

See Also

Chapter 3, "Configuring PeopleSoft CTI," Required Security for PSMCAPI, page 14

Configuring REN Servers

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

This section discusses how to:

- Create REN servers.
- Configure REN servers.
- Define REN servers.
- Determine REN server configuration options.

Selecting 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 up 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 using defaults, which also configure a REN server cluster for each REN server. The default REN server configuration can be changed 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.

This section discusses some possible REN server configurations, dependent 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, DNS addresses the host machine as MachA.mycompany.com, and there is no SSL or reverse proxy server 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

Reverse Proxy Server with 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 **Note.** The cluster URL should never be a secure HTTP address.
REN Server Browser URL	https://MachRPS.mycompany.com:8443 **Note.** This is a secure HTTP address (HTTPS).

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

See Also

Enterprise PeopleTools 8.46 PeopleBook: Security Administration

Page Used to Configure REN Servers

Page Name	Object Name	Navigation	Usage
REN Server Configuration	REN_SERVER_DET_PG	PeopleTools, REN Server Configuration, REN Server Definition	Define a REN server.

Configuring REN Servers

Specify REN server configuration parameters depending 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.
- Specify REN server configuration parameters 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 set in the REN Server and REN Cluster components override any defaults in PSADMIN.

Authentication Domain

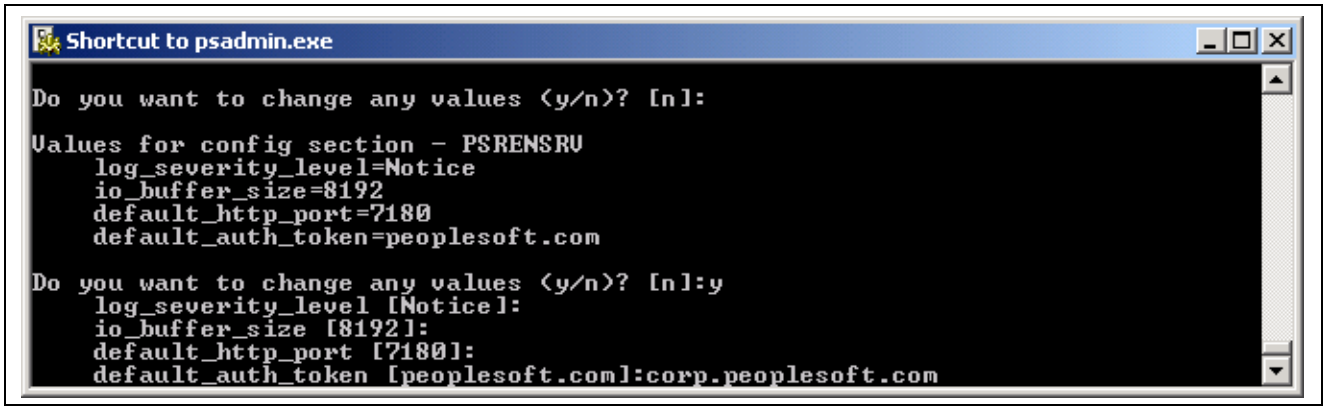
The authentication domain tells PeopleSoft Pure Internet Architecture the internet domain name that browser clients use when accessing PeopleSoft applications across the internet. The token is required to comply with the same-origin security policy that is enforced by most browsers. The domain name specified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

If 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 the REN server and the PeopleSoft Pure Internet Architecture web server are accessed using different DNS names from the browser client (for example, if they are on different machines).

Configuring a REN Server with PSADMIN

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



Configuring PSRENSRV in PSADMIN

Specify parameters as described in the following table:

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

Parameter	Default	Description
default_http_port	7180	The REN server's HTTP port. The default value is *7180*. 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. **Note.** After the HTTP port number 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_auth_token	example.com	The fully qualified domain name of the application server. This value should match the value of the web server's authentication domain. 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. 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.

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

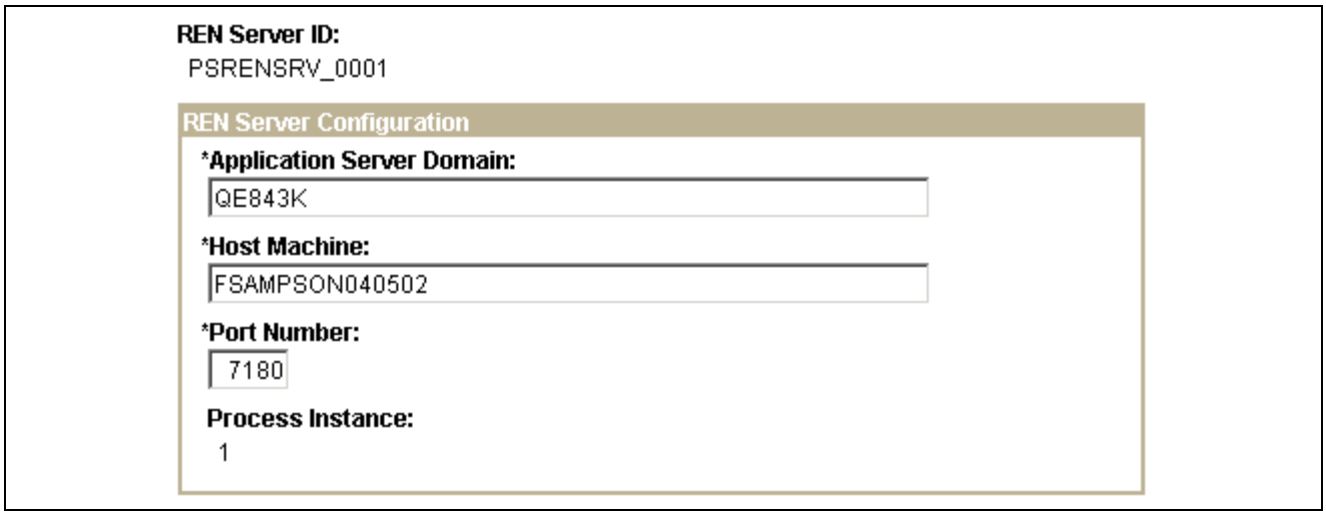
```
set address   192.168.10.1

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.

Defining REN Servers

Access the REN Server Configuration page.



REN Server Configuration page

Application Server Domain	Specify the application server domain serving this REN server.
Host Machine	Specify 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 executing `uname -a`. On a Microsoft Windows machine, determine the host name by executing `hostname` at a command prompt.
Port Number	Specify the HTTP port number on which this REN server is addressed. Change the HTTP port value if multiple REN servers run on the same host machine to avoid port conflicts.
Process Instance	Reserved for future use.

Configuring REN Server Clusters

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

This section discusses how to:

- Cluster REN servers.
- Configure REN server ownership.
- Configure REN server cluster members.

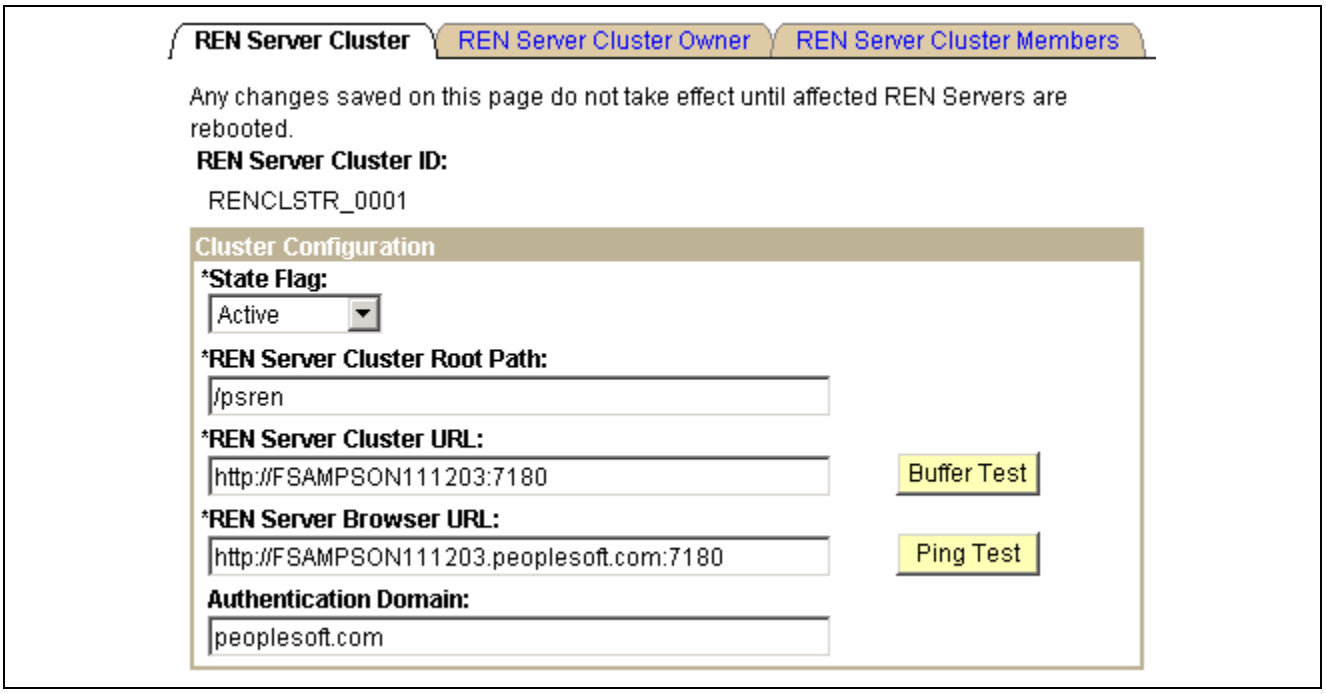
REN server clusters address failover and scalability.

Pages Used to Configure REN Server Clusters

Page Name	Object 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	Define the ownership of the REN server cluster.
REN Server Cluster Members	REN_CLUST_RSERV_PG	PeopleTools, REN Server Configuration, REN Server Cluster, REN Server Cluster Members	Define the REN server clusters member REN servers.

Clustering REN Servers

Access the REN Server Cluster page.



REN Server Cluster page

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

State Flag

Select *Active* or *Inactive.*

This flag 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 an 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.

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 used to reach the REN server cluster internally.

This is the URL 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 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; it cannot 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.

REN Server Browser URL

The REN server browser URL is the address used by external clients and by agent chat to reach the application 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 recognized by your DNS.
- <domain_name> is the full-qualified domain name recognized by your DNS.
- <port> is the REN server port number; the default value is 7180.

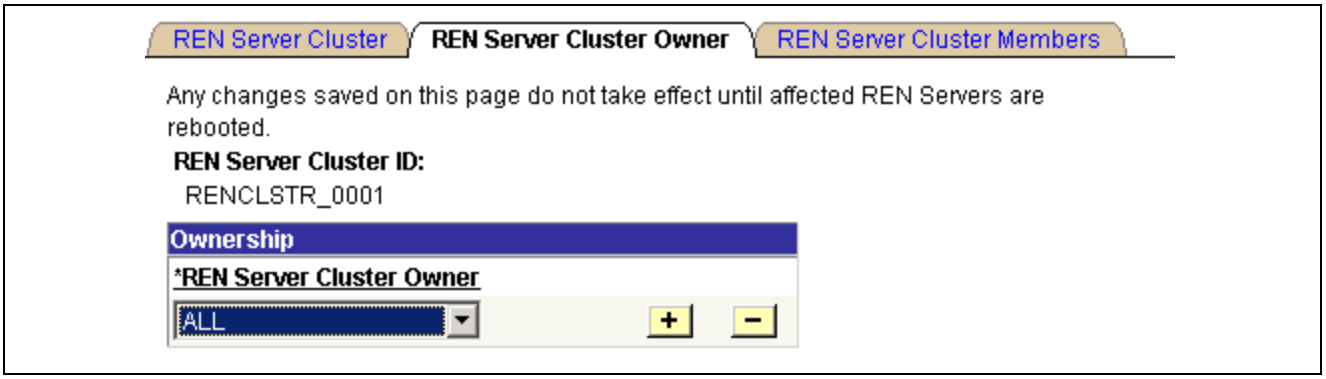
  This port number is the REN server port number or the port number of a proxy server, load balancer, or other front end.

Ping Test Click Ping Test to initiate a test of the REN server 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 that single signon is not implemented.

Authentication Domain Specify the authentication domain. This must be the same as the authentication domain 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.



REN Server Cluster Owner page

REN Server Cluster Owner Select the owner of this REN server cluster from the drop-down list box. Options are:

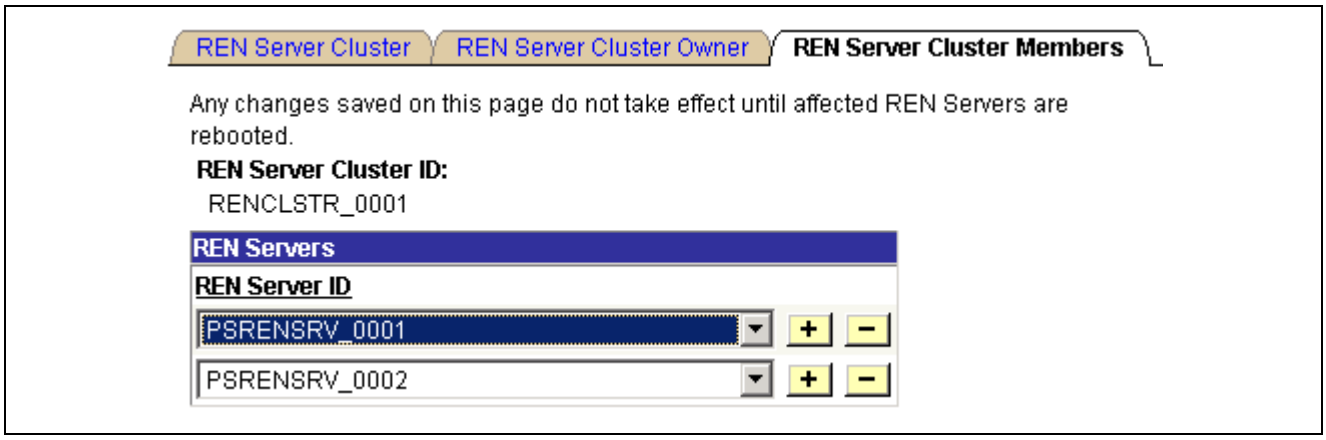
- *All*
- *MCF*
- *Optimization*
- *Reporting*

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 only be created on a REN server cluster owned by MCF or ALL.

Specifying REN Server Cluster Members

Access the REN Server Cluster Members page



REN Server Cluster Members page

REN Server ID Select a REN server from the drop-down list box.

Each REN server can belong to only one REN server cluster.

Example: Configuring a Reverse Proxy Server with a REN Server

This section provides an overview of configuring a reverse proxy server (RPS) for use with one REN server running on separate host machines and provides an example.

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 (the REN server has no native SSL 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.

This example presents one possible configuration for a REN server running on one host machine, and installs an RPS to run on a second host machine, using BEA WebLogic 8.1. The RPS redirects clients to both a REN server and to the PeopleSoft Pure Internet Architecture web server.

This example assumes:

- That you have installed PeopleTools 8.46 on both host machines.
- The you have configured a web server using the default parameters on the first host machine.
- That you have configured a REN server using the default parameters on the first host machine.

Example: Configuring a WebLogic 8.1 RPS for a REN Server on Another Host Machine

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 <PS_HOME>\webserv\rps, and run `startPIA.cmd`.

3. Log on 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*.

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 <PS_HOME>\webserv\rps and run `stopPIA.cmd`.

8. Configure RPS parameters for the *rps* server.

   Locate the file web.xml at PS_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 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>
```

and

```
<servlet-mapping>
 <servlet-name>RENHttpProxyServlet</servlet-name>
 <url-pattern>/psren/*</url-pattern>
</servlet-mapping>
```

9. Reboot the RPS web server.

   Navigate to <PS_HOME>\webserv\rps, and run `startPIA.cmd`.

10. (Optional) Configure and enable SSL on the RPS machine.

CHAPTER 6

Configuring MCF Clusters

This chapter provides an overview of PeopleSoft MultiChannel Framework scalability and failover and discusses how to:

- Configure MCF servers.
- Configure MCF clusters.

Understanding PeopleSoft MultiChannel Framework Scalability and Failover

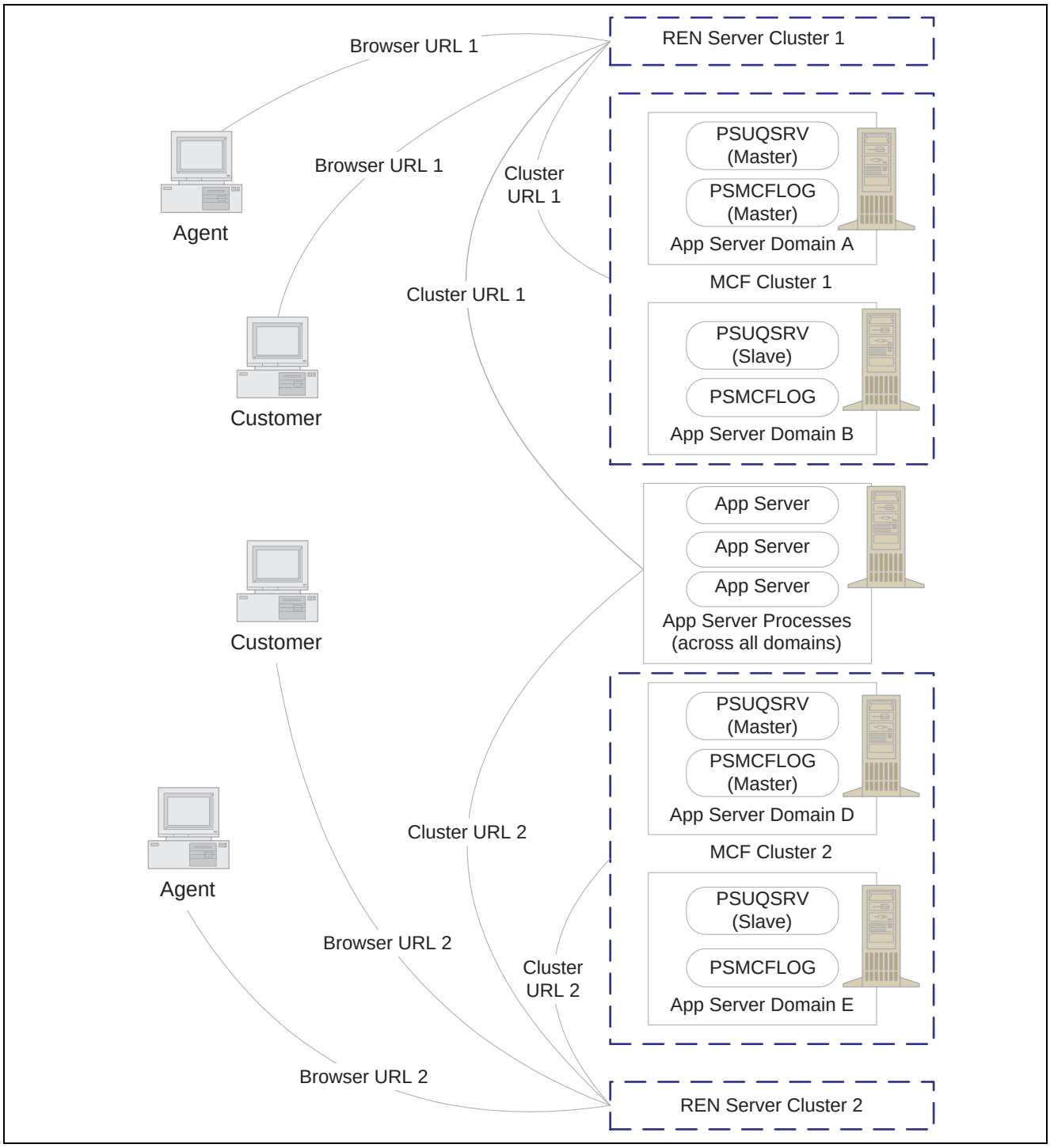
This section discusses:

- PeopleSoft MultiChannel Framework cluster architecture.
- MCF clusters.
- Logical queues and physical queues.
- Queue server failover.
- MCF log server failover.
- MCF cluster configuration.
- Queue server scalability.
- Recommended configurations.

PeopleSoft MultiChannel Framework Cluster Architecture

In a PeopleSoft system, all application server processes—including MCF servers and real-time event notification (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.

This diagram illustrates MCF cluster architecture:



PeopleSoft MultiChannel Framework cluster architecture

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.

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 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 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.

Queue Server Failover

The queue server process (PSUQSRV) is a BEA 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 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 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 MCF Queues,” Tuning Cluster Parameters, page 82.

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.

In summary, configure queue servers to provide hardware failover as follows:

1. Each queue server is part of an MCF cluster.

   To support hardware failover, distribute master and slave queue servers over multiple hosts.

2. Every queue server in an MCF cluster communicates with the same REN server ID.

   Therefore, REN server failover is also crucial.

See Chapter 5, "Configuring REN Servers," page 49.

3. 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. There is no configuration parameter to designate master or slave queue server within a cluster.

4. After an MCF cluster's master queue server is established, all other cluster members become slave queue servers.

5. 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.

   There is no configuration parameter to designate priority among slave servers.

MCF Log Server Failover

The MCF log server (PSMCFLOG) is a BEA Tuxedo–managed server similar to the queue server. Each MCF log server receives events 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 turned off 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 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. There is no specific log server configuration during domain configuration.

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. There is no configuration limit to 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.

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.

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. Since only one physical queue per logical queue can be serviced by a single MCF cluster, this balances load across the servers.

For example, a logical Support queue might be split into physical Support-1 and Support-2 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.

- Use REN server clusters only for failover.

  REN server clusters do not enhance performance.

- Split logical queues into more than one physical queue if there is more work on that queue than a single process or machine can handle.

- If an application server domain is likely to be restarted regularly for reasons 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 MCF Servers

This section provides an overview of MCF server configuration and discusses how to configure MCF servers.

Understanding MCF Server Configuration

PeopleSoft MultiChannel Framework depends on processes 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 utilized by applications 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 combining the machine name, domain subdirectory name, and process identifier. MCF log processes use the same queue server identification scheme.

See Also

Chapter 5, "Configuring REN Servers," page 49

Creating Queue Servers

Create a queue server and log server that start when an application domain is started by selecting MCF Servers from the quick-configure menu during application domain configuration.

Configuring MCF Clusters

To configure MCF clusters, use the MCF Cluster (MCF_UQCLUSTER) component.

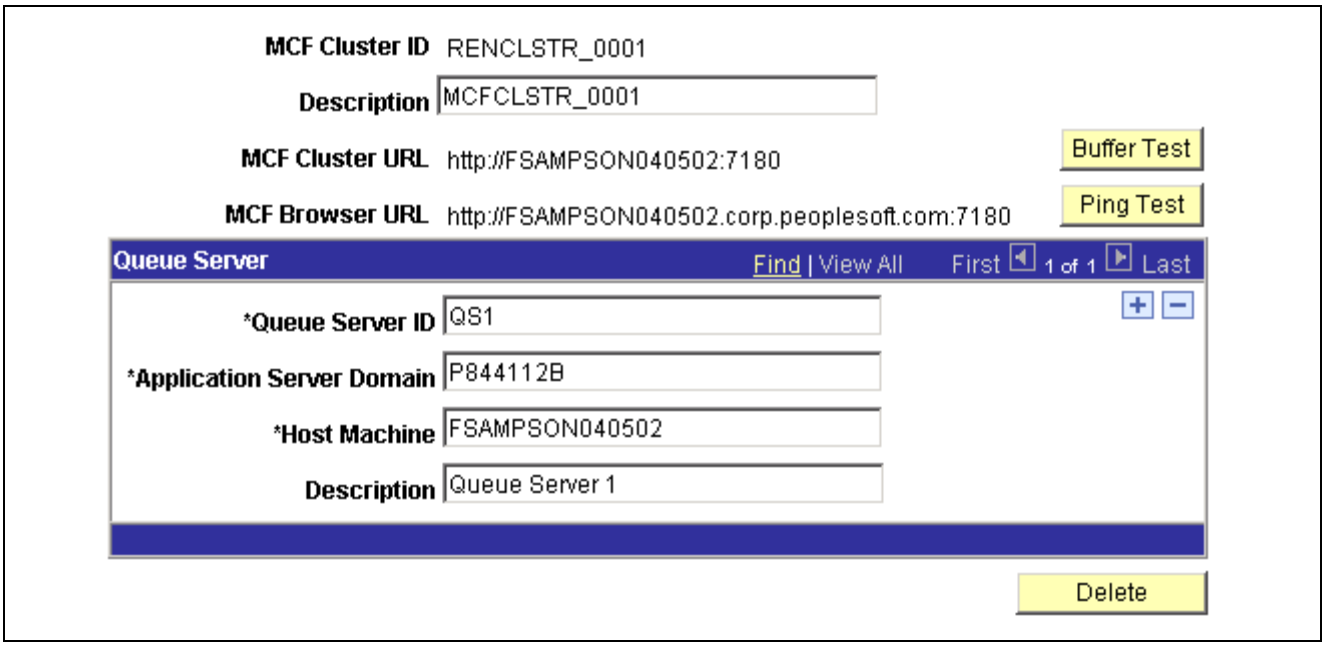
This section discusses how to configure MCF clusters.

Page Used to Configure MCF Clusters

Page Name	Object Name	Navigation	Usage
MCF Cluster	MCF_UQCLUSTER	PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Cluster	Define a cluster of queue servers.

Defining MCF Clusters

Access the MCF Cluster page.



MCF Cluster page

MCF Cluster URL Displays the URL for the REN server that serves this cluster. This is the URL 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 is not dependent 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 the 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 specified in the URL fields. Failure may indicate that a URL is incorrectly specified.

Delete Click to delete the entire MCF cluster. There must not be any active agents or tasks 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, 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 Specify a unique identifier for each queue server to identify its entries in the database control tables.

The log server process paired with this queue server uses this same ID to identify its entry in the log cluster table.

Application Server Domain Specify the application server domain of which this queue server is a member.

Host Machine Specify the name of the application server host machine.

CHAPTER 7

Configuring MCF Queues

This chapter discusses how to:

- Define queues.
- Configure tasks.
- View the cluster summary page.
- Tune cluster parameters.
- Notify clusters of changed parameters.

See Also

Enterprise PeopleTools 8.46 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:

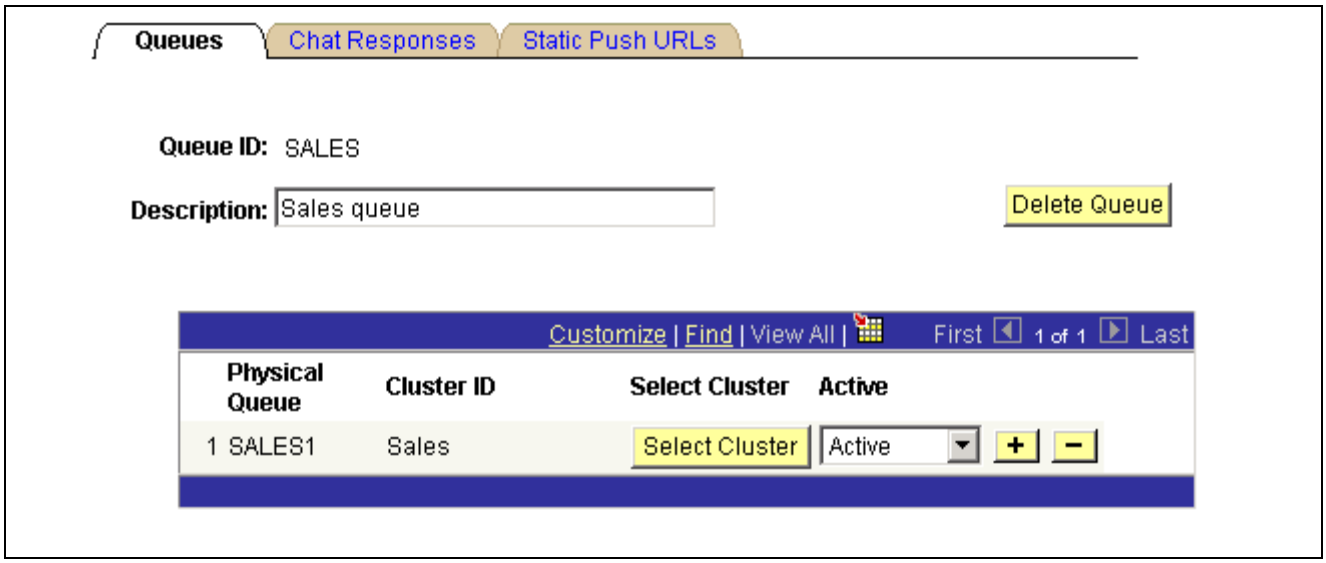
- Define queues.
- Define chat responses.
- Define static push URLs.

Pages Used to Define Queues

Page Name	Object Name	Navigation	Usage
Queues	MCF_QUEUE_PG	PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Queue, Queue	Add a queue.
Chat Responses	MCFSYSMSG_PG	PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Queue, Chat Responses	Define 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.



Queues page

Queue ID Queue IDs must be alphanumeric, may not end in a numeral, but may 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 assigned to the child physical queue will receive a message notifying them to log off of their MultiChannel Consoles when the logical queue is deleted.

Physical Queue

The Physical Queue field uniquely identifies that part of a logical queue 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 added.

Physical queue IDs are automatically generated. The maximum number of physical queues is nine.

Cluster ID

The Cluster ID field identifies the MCF cluster that services this physical queue. Click Select Cluster to choose from configured MCF clusters.

Each MCF cluster can only service 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 Select Cluster to choose a cluster from a list of available MCF clusters.

The selected MCF cluster will service this physical queue. The primary queue server in this cluster will manage tasks and agents assigned to this physical queue.

Active

Select *Active* or *Inactive* from the drop-down list box.

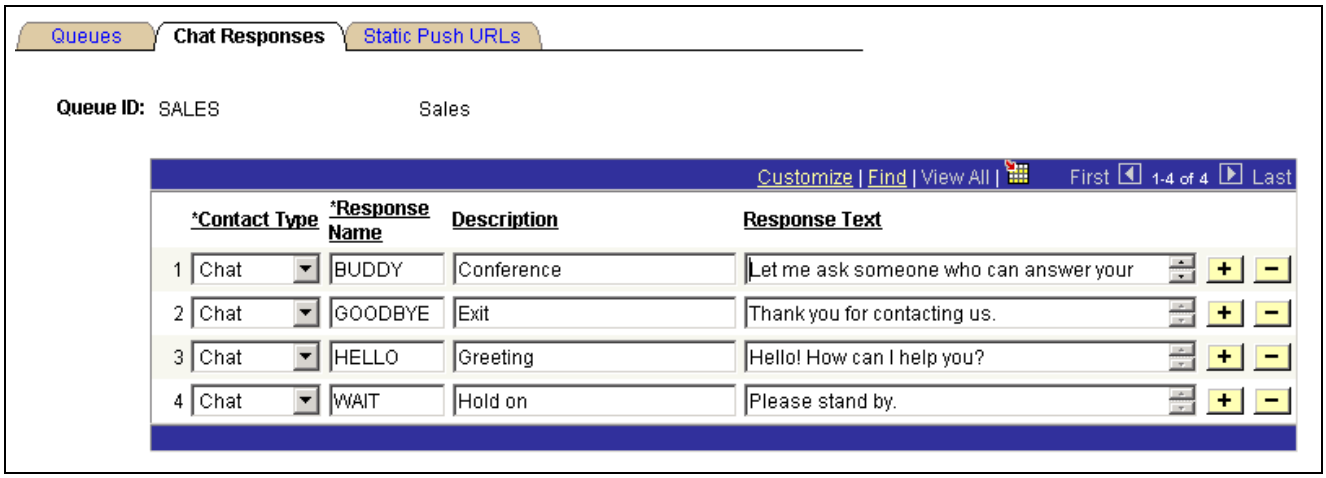
The queue server will not send new tasks to an inactive physical queue. Agents and existing tasks will remain on an inactive physical queue. The physical queue must be active to receive new queued tasks.

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, 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 Chat Responses

Access the Chat Responses page.



Chat Responses page

Chat responses specified on this page are available to all agents logged on to this queue.

Contact Type Specify one of the following contact types:

- *Chat*
- *Email* (not currently supported)
- *Generic* (not currently supported)

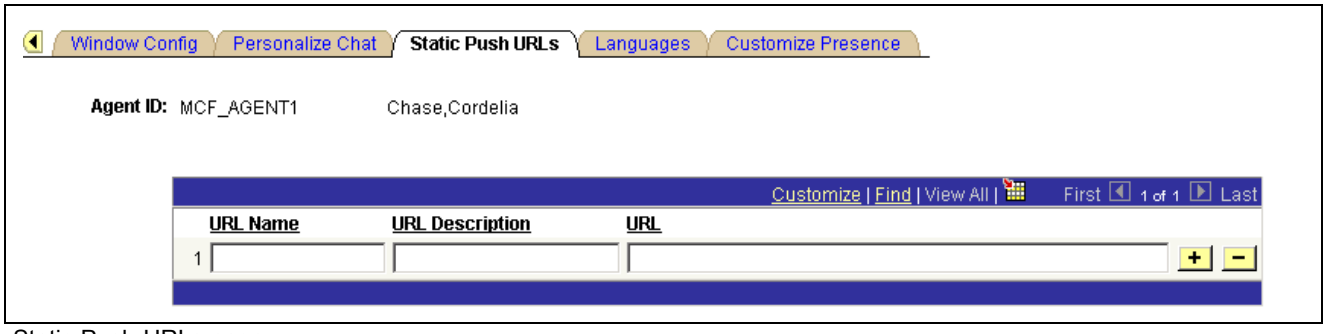
Response Name The response name appears in the agent's Template Messages drop-down list box for all agents who belong to a physical queue on this logical queue

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.



Static Push URLs page

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 Specify a name to identify the URL.

The URL name appears in the agent's Static URL drop-down list box for all agents that belong to this logical queue

URL Description Specify 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 Specify the queue push URL.

The URL must include the opening http:// and any required parameters.

All static URLs 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, be sure 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.

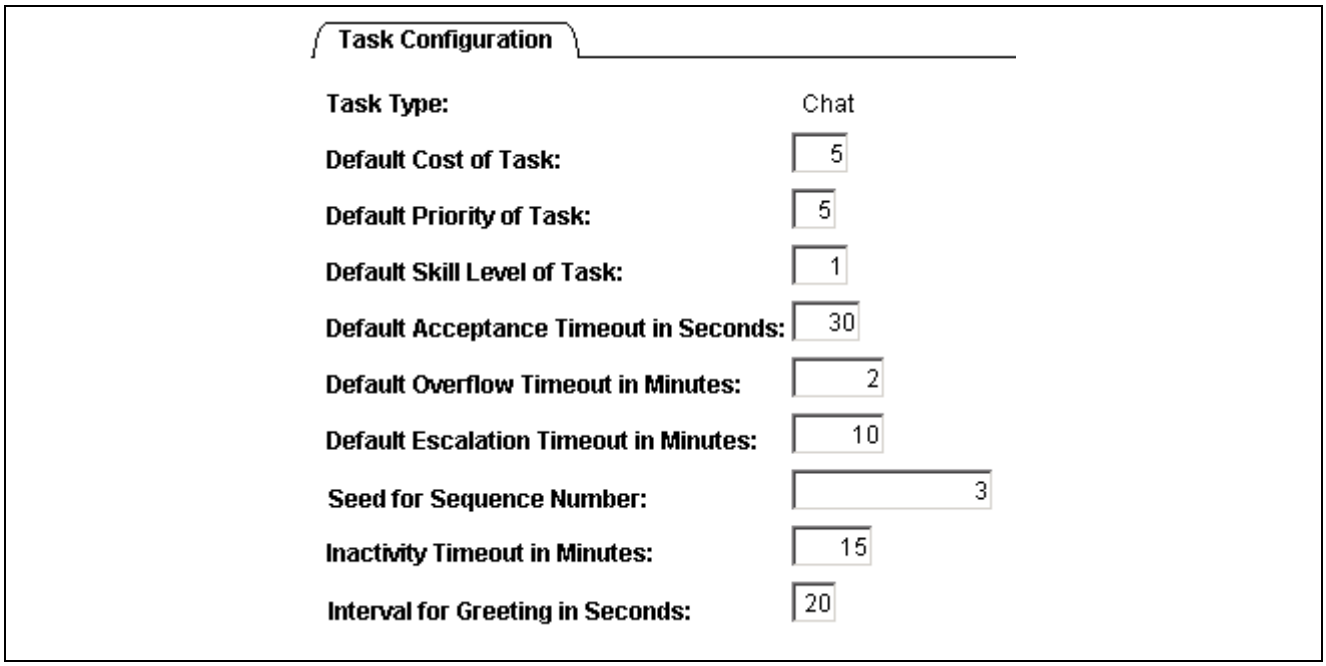
This section discusses how to configure tasks.

Page Used to Configure Tasks

Page Name	Object Name	Navigation	Usage
Task Configuration	MCF_TASKCFG_PG	PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Tasks	Configure tasks such as chat, email, voice, and generic alerts.

Configuring Tasks

Access the Task Configuration page.



Task Configuration page

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.

Default Cost of Task Specify the cost of the task.

Cost is a measure of the workload each task places on an agent. The cost of a task is an estimate of the task's expected complexity and of the time required to resolve the task. The minimum value is *0,* and there is no maximum value.

The costs of tasks assigned to an agent are added up and evaluated against the maximum workload for each agent to determine if 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 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*

Default Priority of Task Specify 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 there is no maximum value.

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 will always assign 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 will be assigned in the order of their enqueue time.

The value 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

Specify the minimum agent skill required to handle this task.

The queue server assigns this task type to an available agent with the lowest skill level on that queue greater than or equal to the skill level required by the task.

The minimum value is *0,* and there is no maximum value.

The value 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 (not currently supported): *2*

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 re-enqueued for assignment to another agent.

The queue server uses an algorithm to minimize reassignment of tasks which previously timed out to the same agent.

The value 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 (not currently supported): *10*

Default Overflow Timeout in Minutes

Specify the overflow timeout.

The overflow timeout is the time period that a queue server has to find an agent who accepts a task (click 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 specified here can be overridden in the EnQueue() or InitChat() built-in function call.

Default overflow timeouts are:

- Chat: *2*
- Email: *120*
- Generic: *120*
- Voice (not currently supported): *1*

Default Escalation Timeout in Minutes

Specify the default escalation timeout.

The escalation timeout 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 timeout is only valid on a chat session after it is accepted by an agent, and has no effect on voice tasks (CTI).

The value specified here can be overridden in the EnQueue() or InitChat() built-in function call.

Default escalation timeouts:

- Chat: *10*
- Email: *480*
- Generic: *480*
- Voice (not currently supported): *2*

Seed for Sequence Number

This field displays the current cumulative count of tasks of this type 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 timeout

Inactivity timeout 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 appropriate to your business requirements. For example, task cost could vary over a range of 1 to 100 instead of 1 to 10.

See Also

Chapter 8, “Configuring MCF Agents,” Creating Agents, page 89

Viewing the Cluster Summary

To view the Cluster Summary page, use the Cluster Summary (MCF_RSERV_CFG_CMP) component.

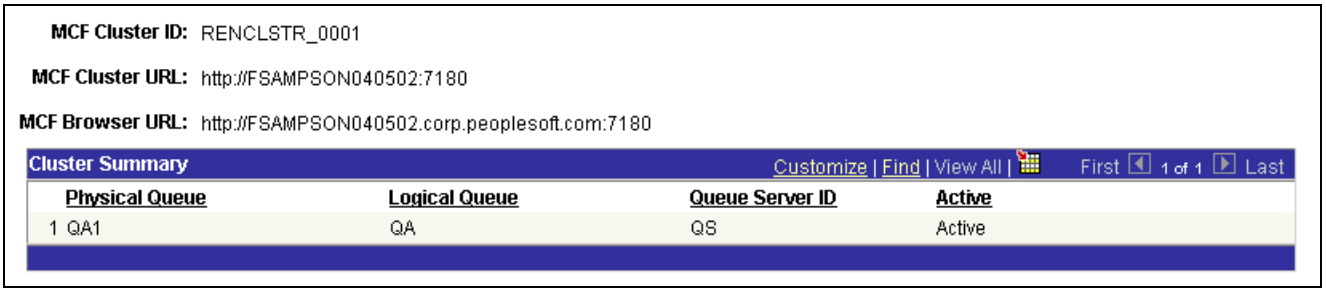
This section discusses how to view the Cluster Summary page.

Page Used to View the Cluster Summary

Page Name	Object 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.



Cluster Summary page

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.

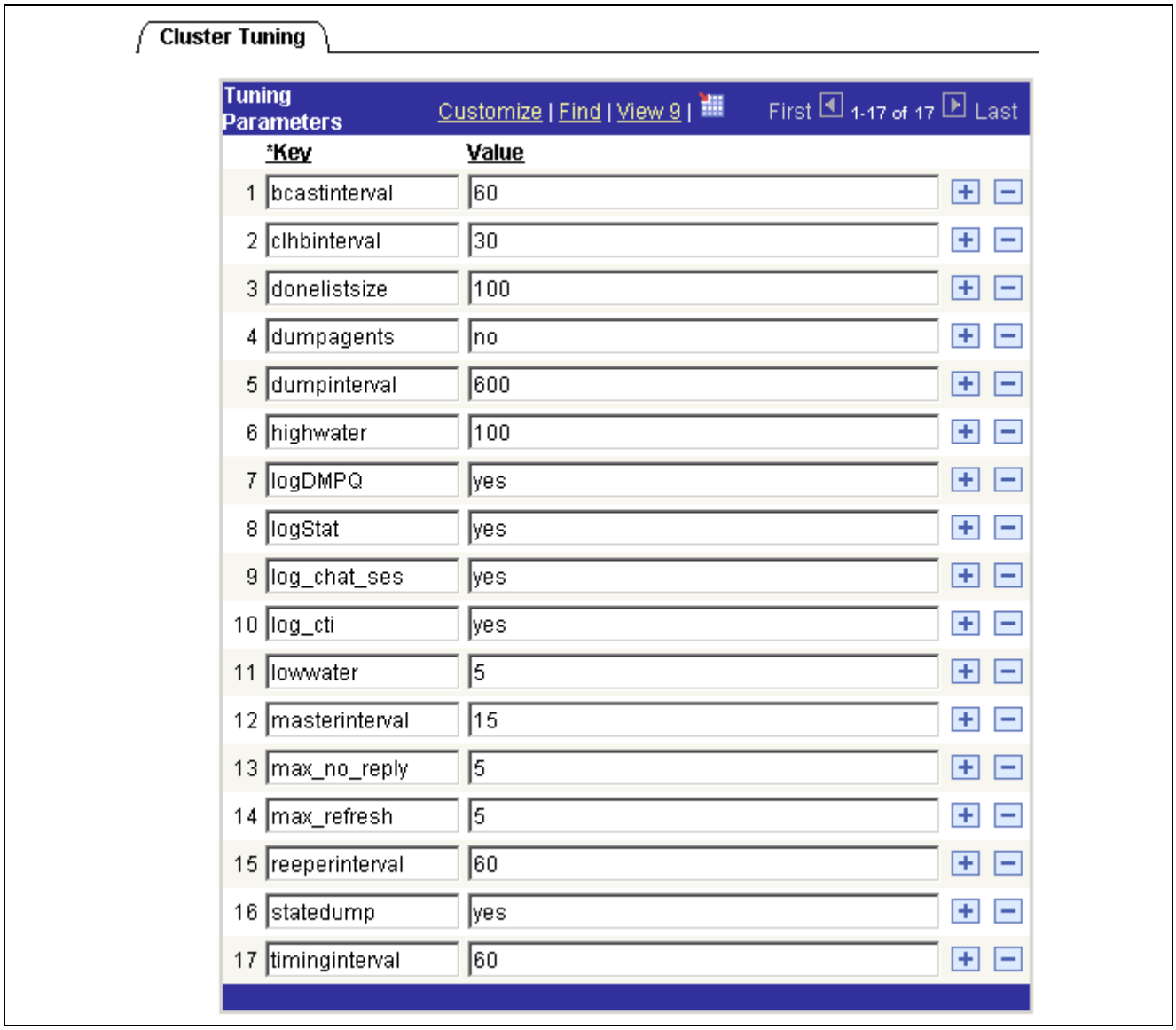
This section discusses how to tune cluster parameters.

Page Used to Tune Clusters

Page Name	Object 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.



Cluster Tuning page

Use the Cluster Tuning page to set MCF cluster parameters to optimize performance or enable logging according to the following table:

Key	Default value	Usage
bcastinterval	*60*	Interval, in seconds, after which the number of unassigned tasks per physical queue and the number of agents logged into each physical queue are broadcast to the MultiChannel Consoles for display next to the queue names. 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. The bcastinterval value also determines how frequently onStat2 event statistics are calculated. A smaller value provides updated statistics more frequently. See Chapter 11, "Developing for PeopleSoft MultiChannel Framework," Using the Monitor Queues Page and Sample Monitor - Queue Statistics Page, page 139. See Chapter 11, "Developing for PeopleSoft MultiChannel Framework," Using the Monitor Agents Page and Sample Monitor - Agent States Page, page 137.
clhbinterval	*30*	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. See Chapter 11, "Developing for PeopleSoft MultiChannel Framework," Using and Demonstrating JSMCAPI, page 129.
donelistsize	*100*	Number of completed tasks stored in the list that is used to calculate average task duration. Configure the donelistsize value depending on the task volume encountered and the interval over which you want to monitor tasks. See Chapter 11, "Developing for PeopleSoft MultiChannel Framework," Using and Demonstrating JSMCAPI, page 129.
dumpagents	*No*	Specify *Yes* if the status of agent activity should be written to the database during the periodic state dumps. Logging agent status increases queue server load, but provides information on agent performance. See Chapter 9, "Administering PeopleSoft MultiChannel Framework," Viewing the Agent State Summary, page 104.

Key	Default value	Usage
dumpinterval	*600*	Interval, in seconds, after which the queue state is written to the database. 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. A value of less than one minute will significantly reduce queue server performance. See Chapter 9, "Administering PeopleSoft MultiChannel Framework," Viewing the Queue Server State, page 103. See Chapter 9, "Administering PeopleSoft MultiChannel Framework," Viewing the Queue State Summary, page 104.
highwater	*100*	Maximum number of persistent tasks retrieved from the database and cached in memory in the queue server. The high- and lowwater mark values determine how often, and how many, persistent tasks should be read into memory. 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. 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.
logDMPQ	*No*	Specify *Yes* if you want PSMCFLOG to log REN server event notifications resulting from bcastinterval broadcasts. Only the event is logged, not its contents. See Chapter 9, "Administering PeopleSoft MultiChannel Framework," Viewing Event Logs, page 107.
logStat	*No*	Specify *Yes* to log the statistics returned by the queue server for the onStat1 user and group events to the database. See Chapter 11, "Developing for PeopleSoft MultiChannel Framework," Using and Demonstrating JSMCAPI, page 129.

Key	Default value	Usage
log_chat_ses	*No*	Specify *Yes* to turn on logging of the contents of chat sessions. See Chapter 9, "Administering PeopleSoft MultiChannel Framework," Viewing Chat Logs, page 106.
log_cti	*No*	Specify *Yes* to turn on logging of CTI events. See Chapter 3, "Configuring PeopleSoft CTI," CTI Event Logging, page 35.
lowwater	*5*	Minimum number of persistent tasks 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. The high- and lowwater mark values determine when, and how many, persistent tasks should be read into memory. 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. PeopleSoft recommends that the lowwater value be greater than or equal to the maximum number of agents logged onto any physical queue at one time.
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. 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. If a only one queue server is configured for an MCF cluster, this value can be large. The masterinterval value also acts as a heartbeat interval for the master queue server connection to user consoles.
max_no_reply	*5*	Sets the maximum number of consecutive agent timeouts before the queue server automatically logs out the agent and sets the agent's console status as Assumed Unavailable.
max_refresh	*5*	Sets the maximum number of consecutive times that results are discarded when refreshing the task queue from the database if there is an intervening notification of new persistent tasks.

Key	Default value	Usage
reeperinterval	*60*	Interval, in seconds, after which deleted task are cleared from memory in the queue server. A lower value increases queue server load but clears memory more frequently. A higher value decreases queue server load but clears memory less frequently.
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. See Chapter 11, "Developing for PeopleSoft MultiChannel Framework," Using and Demonstrating JSMCAPI, page 129.
timinginterval	*60*	Interval, in seconds, after which the database is checked for expired or overflowed persistent tasks. This parameter does not affect real-time tasks. 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.

Notifying Clusters of Changed Parameters

To notify clusters of changed parameters, use the Cluster Notify (MCF_AD_NOTIFY_CMP) component.

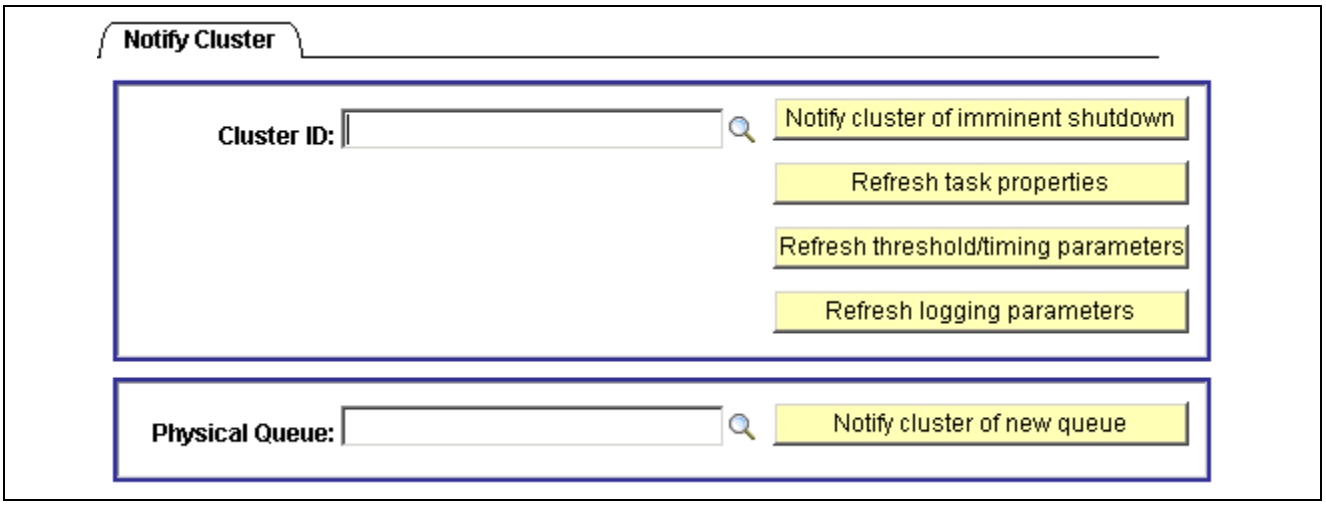
This section discusses how to notify clusters of changed parameters.

Page Used to Notify Clusters of Changed Parameters

Page Name	Object Name	Navigation	Usage
Notify Cluster	MCF_CL_NOTIFY_PG	PeopleTools, MultiChannel Framework, Universal Queue, Configuration, Cluster Notify	Notify a cluster of changes to parameters, configuration, or of shutdown.

Notifying Clusters of Changed Parameters

Access the Notify Cluster page.



Notify Cluster page

Use the Notify Cluster page to notify an MCF cluster of certain changes to its parameters, 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 every agent logged on to the selected MCF 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 MCF cluster.

Refresh threshold/tuning parameters Click to reload parameters, other than logging parameters, changed on the Cluster Tuning page for the selected MCF cluster.

Refresh logging parameters Click to reload logging parameters 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.

CHAPTER 8

Configuring MCF Agents

This chapter discusses:

- Defining agents.
- Defining 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.
- Customize an agent's presence.

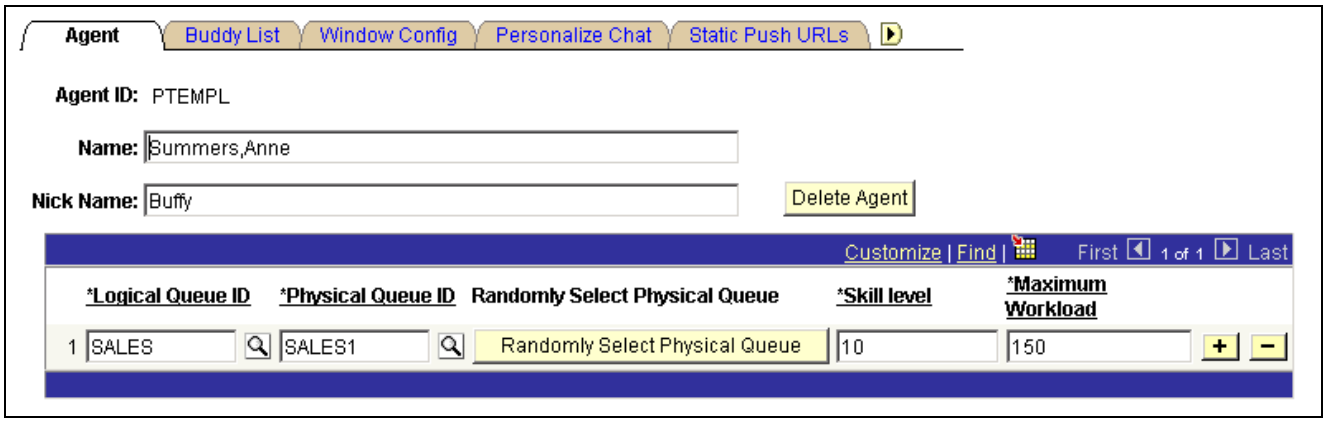
These three agent definition pages form the basis of agent configuration. Other parameters are optional.

Pages Used to Define Agents

Page Name	Object Name	Navigation	Usage
Agent	MCF_AGENT_PG	PeopleTools, MultiChannel Framework, Universal Queue, Administration, Agent, Agent	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.

Creating Agents

Access the Agent page.



Agent page

Name Enter the full name, in PeopleSoft format, of this agent.

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.

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 may 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 assignment.

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 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 will help to spread multiple agents evenly over available physical queues.

Skill Level Select the skill level of this agent for the tasks assigned for this queue. This is a required field.

The agent is assigned only tasks requiring a skill level less than or equal to the skill level specified here. If there is more than one qualified agent 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.

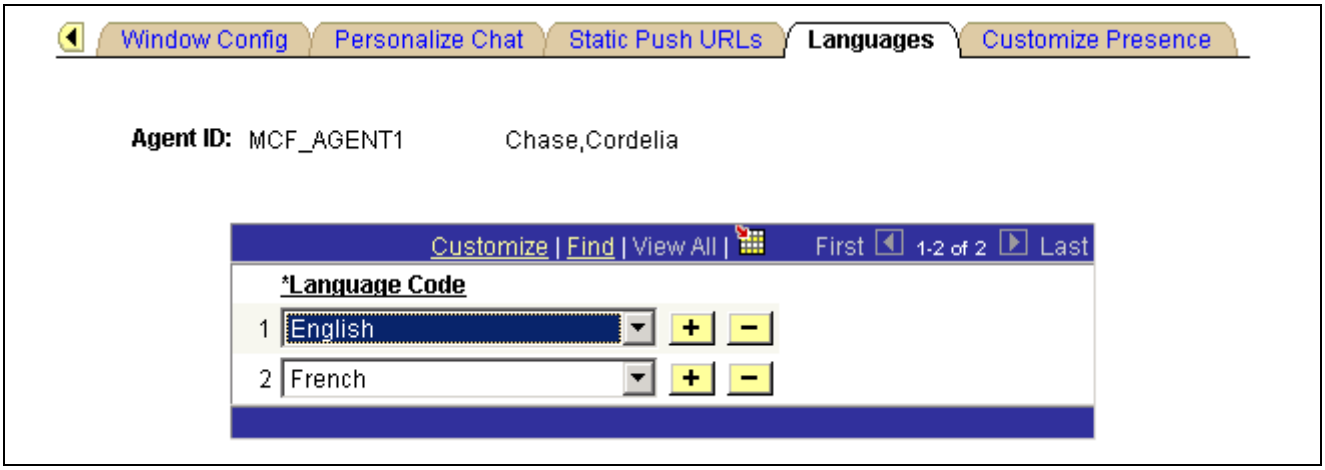
Maximum Workload Select the maximum load that this agent can be assigned before tasks are held or assigned to other agents. This is a required field.

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 on that queue.

Specifying Languages That an Agent Supports

Access the Languages page.



Languages page

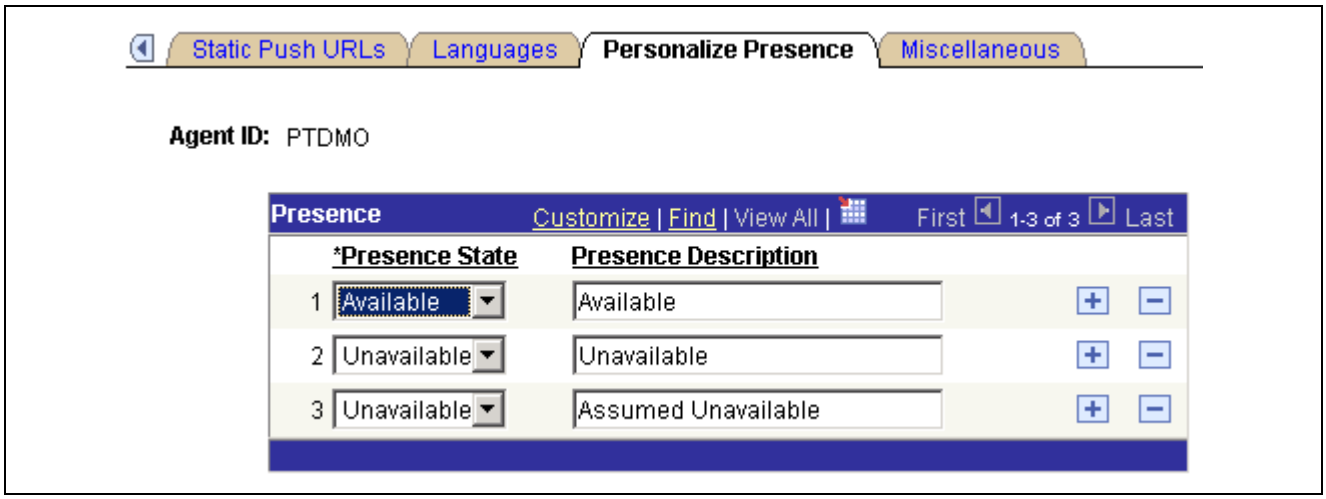
Specify the languages that each agent is qualified to support.

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.



Personalize Presence page

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 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.

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.
- Configure windows.
- Personalize chat.
- Specify agent-specific URLs.
- Specify miscellaneous parameters.

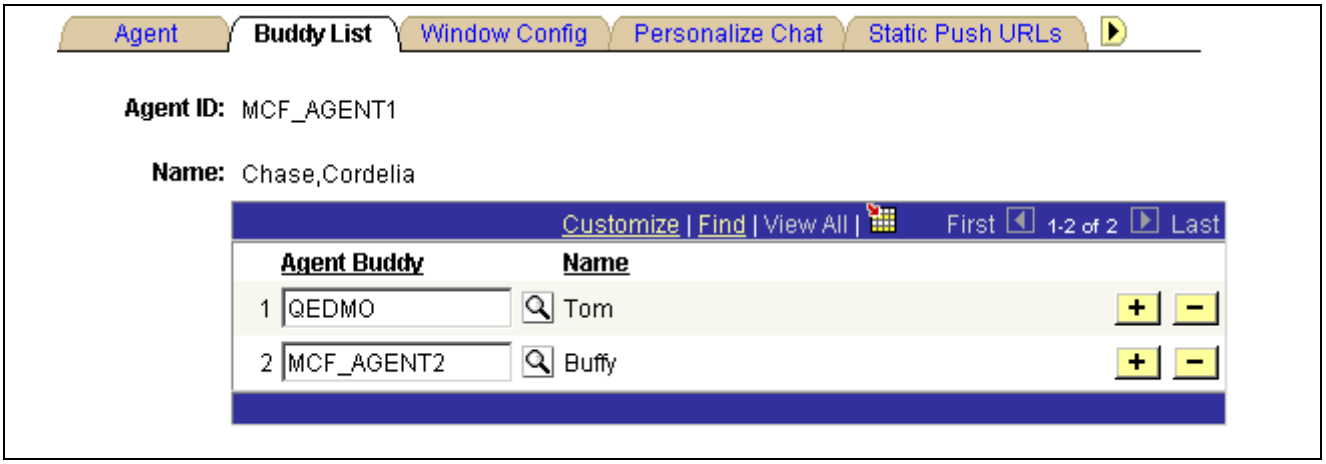
These agent configuration pages are considered optional because most do not have default values and can remain unconfigured without affecting an agent's ability to log on to a queue and accept tasks.

Pages Used to Define Optional Agent Characteristics

Page Name	Object 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.



Buddy List page

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.

Each agent buddy must be logged onto 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 onto physical queues on the same MCF cluster.

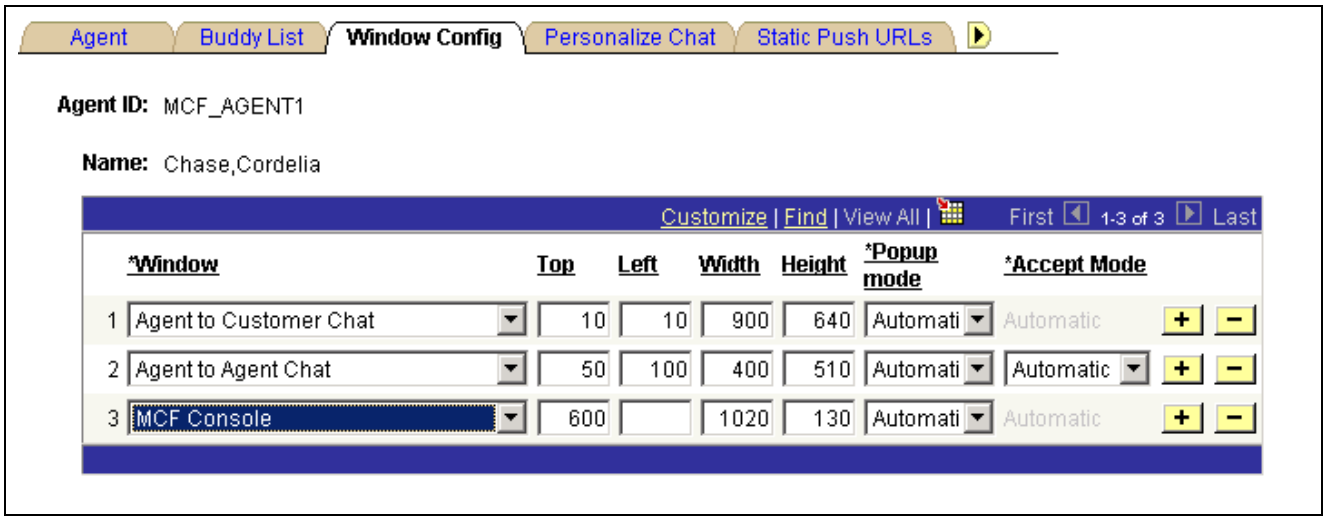
Name Displays the buddy agent's nickname.

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.

Configuring Windows

Access the Window Config page.



Window Config (window configuration) page

Set the initial agent window placement and size by specifying parameters on this page. The windows may be resized and moved by an agent.

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.

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:

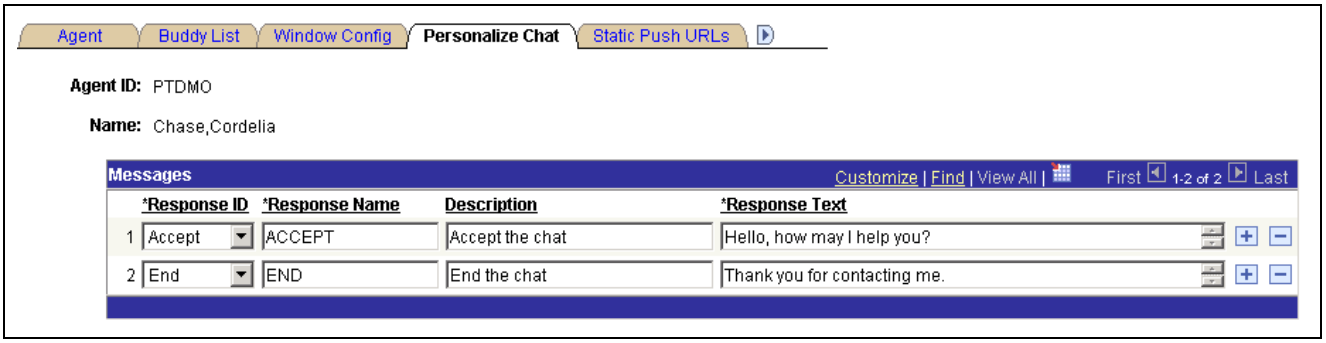
Automatic: Agent-to-agent chats are automatically accepted without requiring the agent to click the icon.

*Manual:*Agent-to-agent chats require the agent to click the icon.

Accept mode affects only collaborative chat.

Personalizing Chat

Access the Personalize Chat page.



Personalize Chat page

An agent can create personalized responses in addition to the system responses defined for each queue.

Response ID

Responses, except those identified by 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 message appears when the agent accepts the abandoned chat.
- *Accept:*This greeting 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.

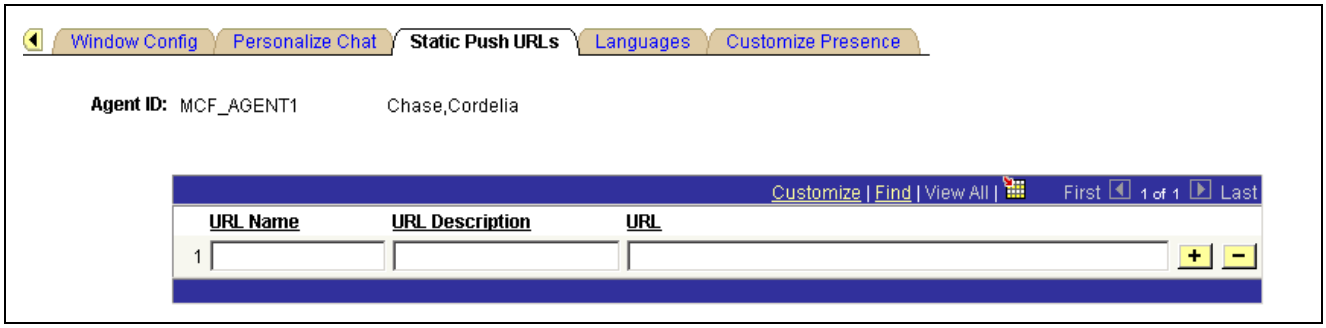
- *Answer Question:*This greeting is automatically sent to the chat initiator when an agent accepts a chat in response to a chat request that includes a question.
- *Deny:*Only applies to collaborative chat. If an agent elects not to accept a chat, this message is automatically sent to the chat initiator.
- *End:* If either party exits 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) 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.



Static Push URLs page

This page defines URLs that this agent can send to a client browser. These URLs are in addition to the URLs 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.

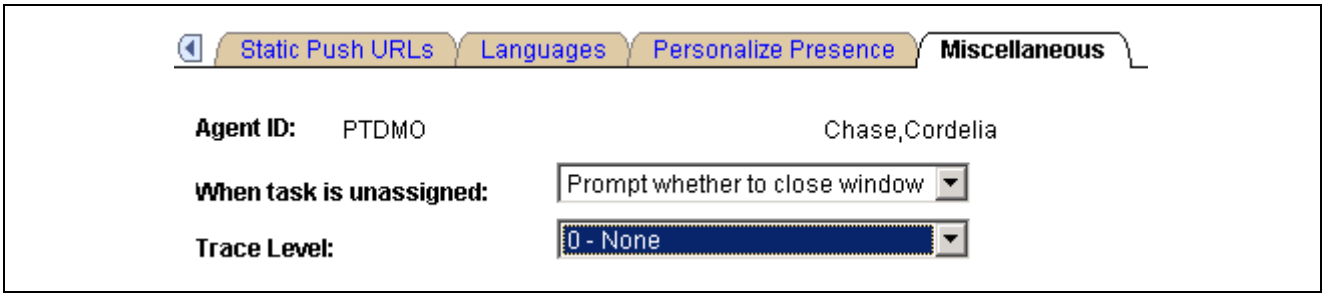
The URL must include the opening http:// and any required parameters.

All static URLs 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.



Miscellaneous page

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*

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.

CHAPTER 9

Administering PeopleSoft MultiChannel Framework

This chapter discusses how to:

- Administer physical queues.
- View queue server, queue, and agent state.
- View 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:

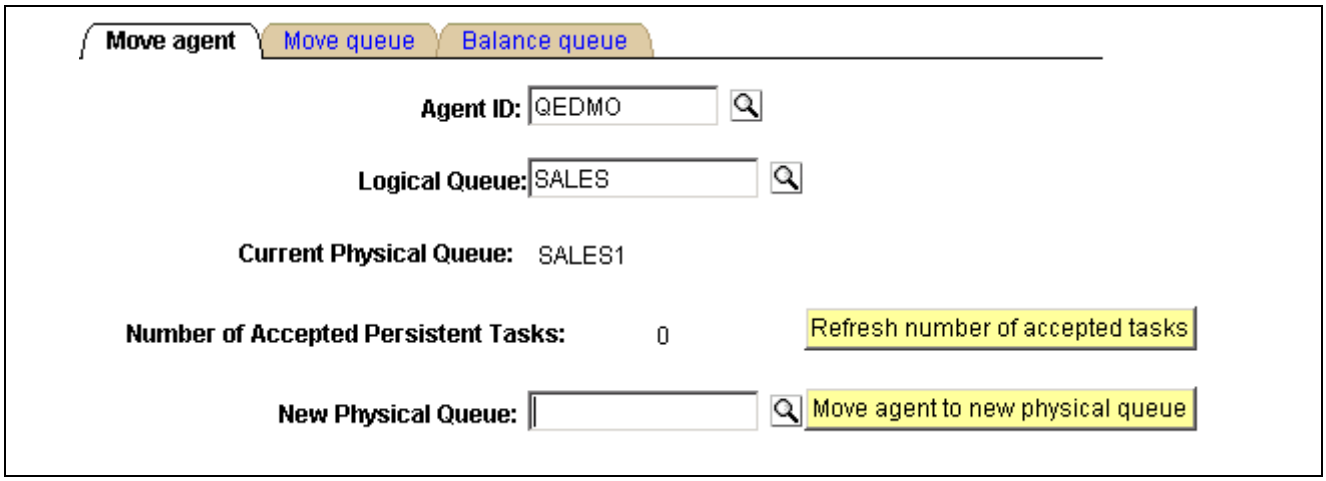
- Move agents between physical queues.
- Move queues.
- Balance queues.

Pages Used to Administer Physical Queues

Page Name	Object 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.
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.



Move agent page

You can move an agent, and any open persistent tasks that are associated with that agent, from one physical queue to another physical queue on the same logical queue. For example, if an agent is transferred to another office, you can move the agent to another physical queue on a server closer to the new office.

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.

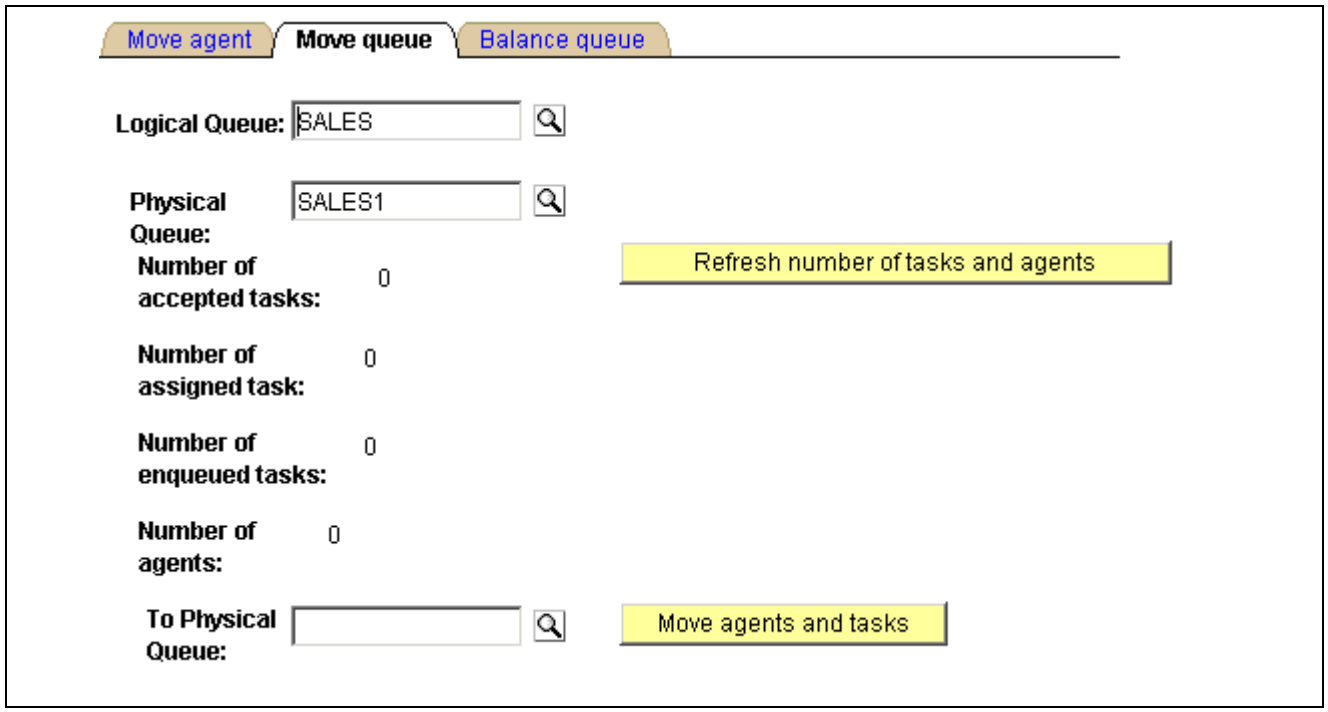
Refresh number of accepted tasks Click to update the number of persistent tasks accepted by this agent on the current physical queue.

New Physical Queue	Specify 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.

Note. Ongoing chat sessions, which are not persistent tasks, are not affected by the move agent action.

Moving Queues

Access the Move queue page.



Move queue page

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 tasksand**Number of agents**	Displays the number of enqueued tasks and assigned agents on this physical queue.
To Physical Queue	Specify 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.

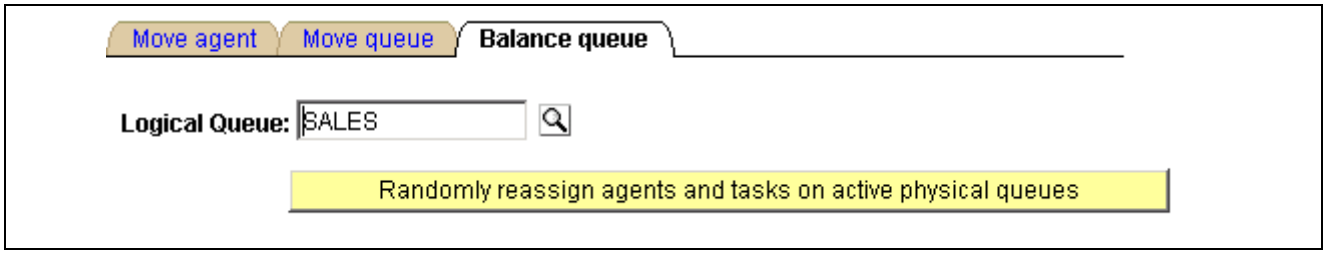
Note. The physical queue must be inactive before moving agents and tasks. Inactivate the physical queue on the Queues page. Agents on the inactive physical queue are automatically logged off before being moved.

See Also

Chapter 7, "Configuring MCF Queues," Defining Queues, page 73

Balancing Queues

Access the Balance queue page.



Balance queue page

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.

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:

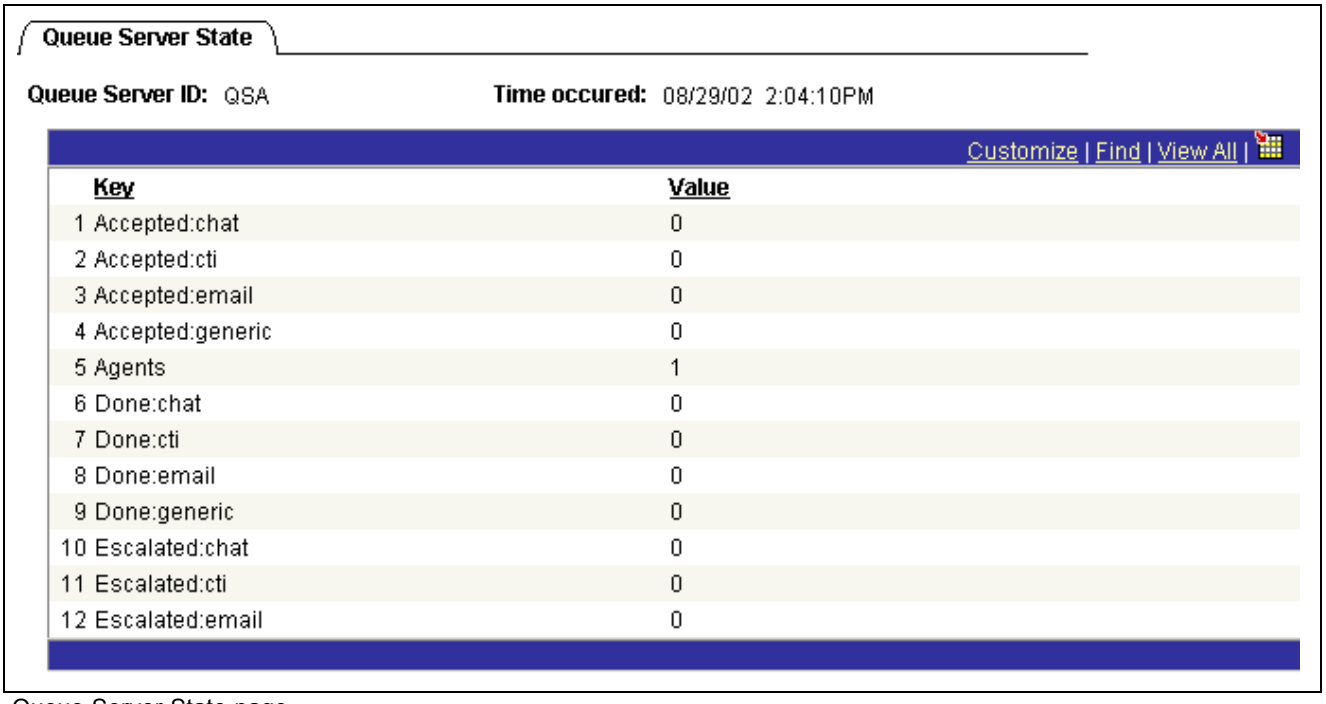
- View the queue server state.
- View the queue state summary.
- View the agent state summary.

Pages Used to View Queue Server, Queue, and Agent States

Page Name	Object 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.



Queue Server State page

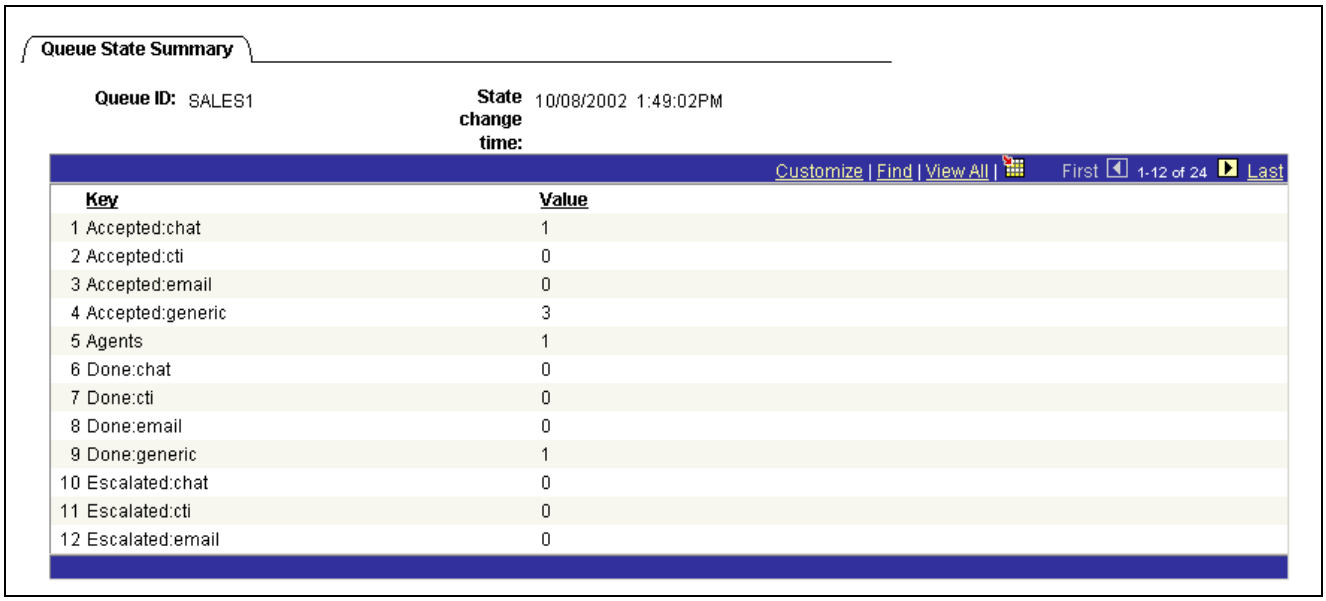
The Queue Server State page displays cumulative diagnostic totals for the selected queue server, at the selected time, across all physical queues serviced by this queue server. Cumulative totals are reset after each state dump.

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 very 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. The intervals are configured on the Cluster Tuning page.

Viewing the Queue State Summary

Access the Queue State Summary page.



Queue State Summary page

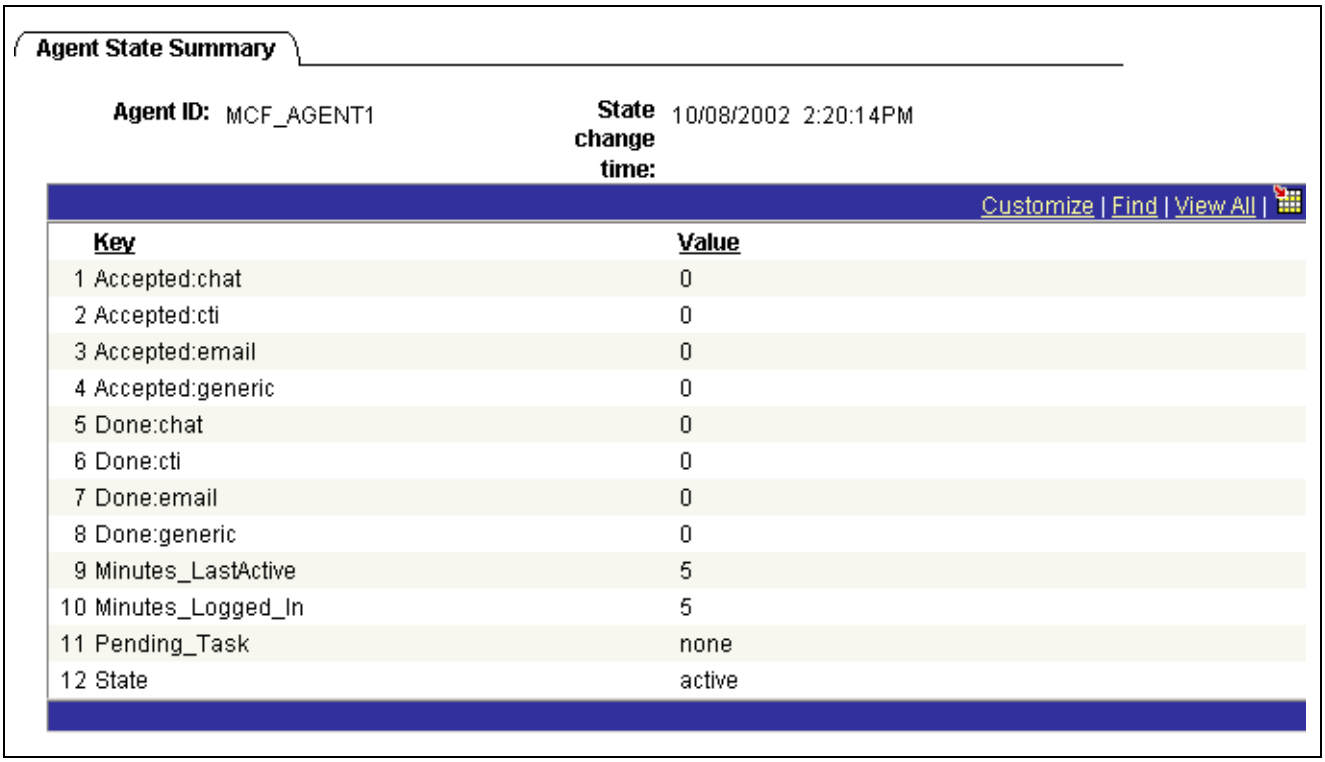
The Queue State Summary page displays cumulative diagnostic totals for the selected physical queue.

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 very 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.



Agent State Summary page

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 affect performance.

See Also

Chapter 7, “Configuring MCF Queues,” Tuning Cluster Parameters, page 81

Viewing Chat and Event Logs

To view chat and event logs, use the Chat Log (MCF_CHAT_LOG_CMP) and Event Log (MCF_EVENTLOGL_CMP) components.

This section discusses how to:

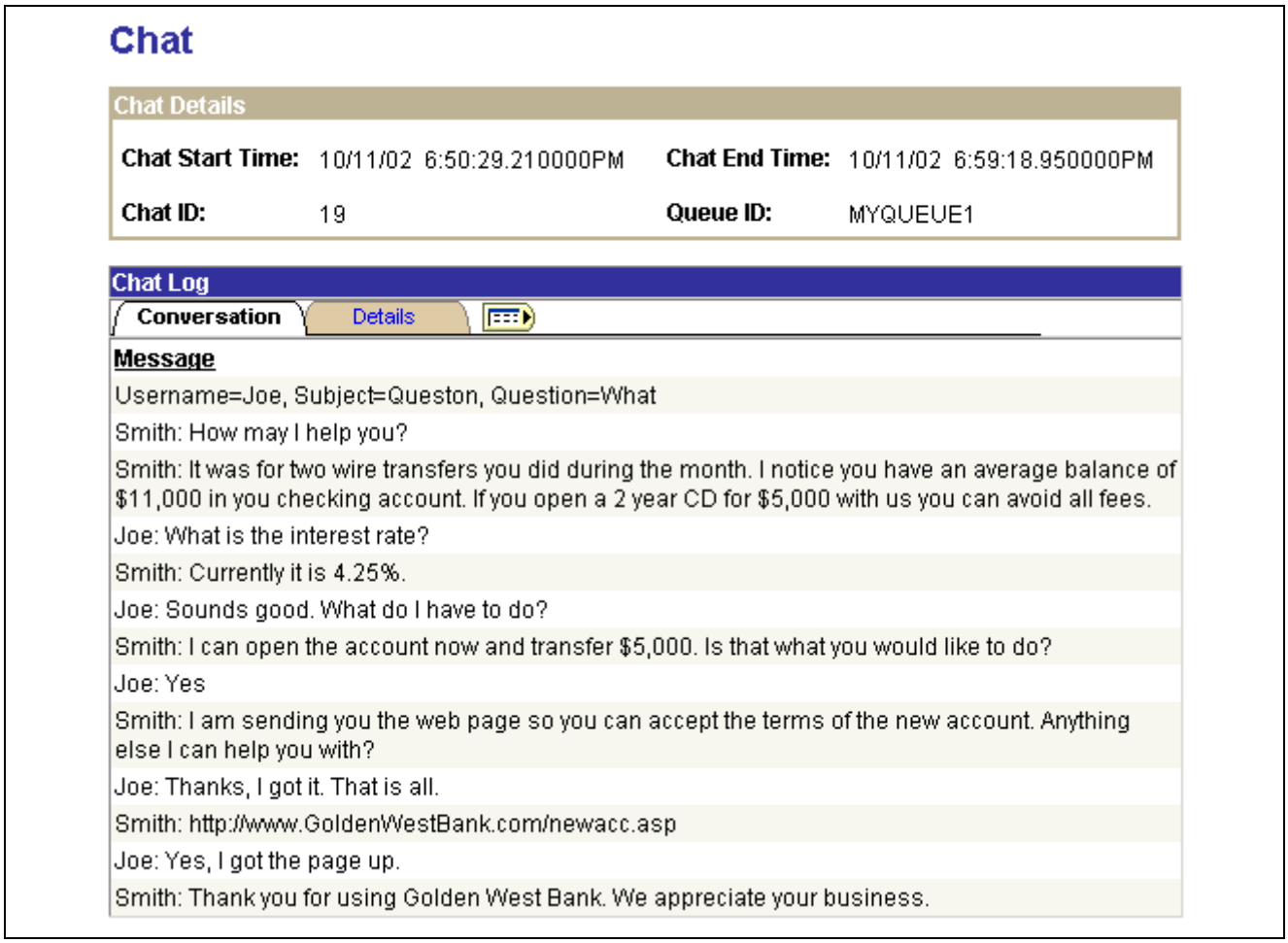
- View chat.
- View event logs.
- View MCF logs.

Pages Used to View Chat and Event Logs

Page Name	Object Name	Navigation	Usage
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 Chat Logs

Access the Chat page.



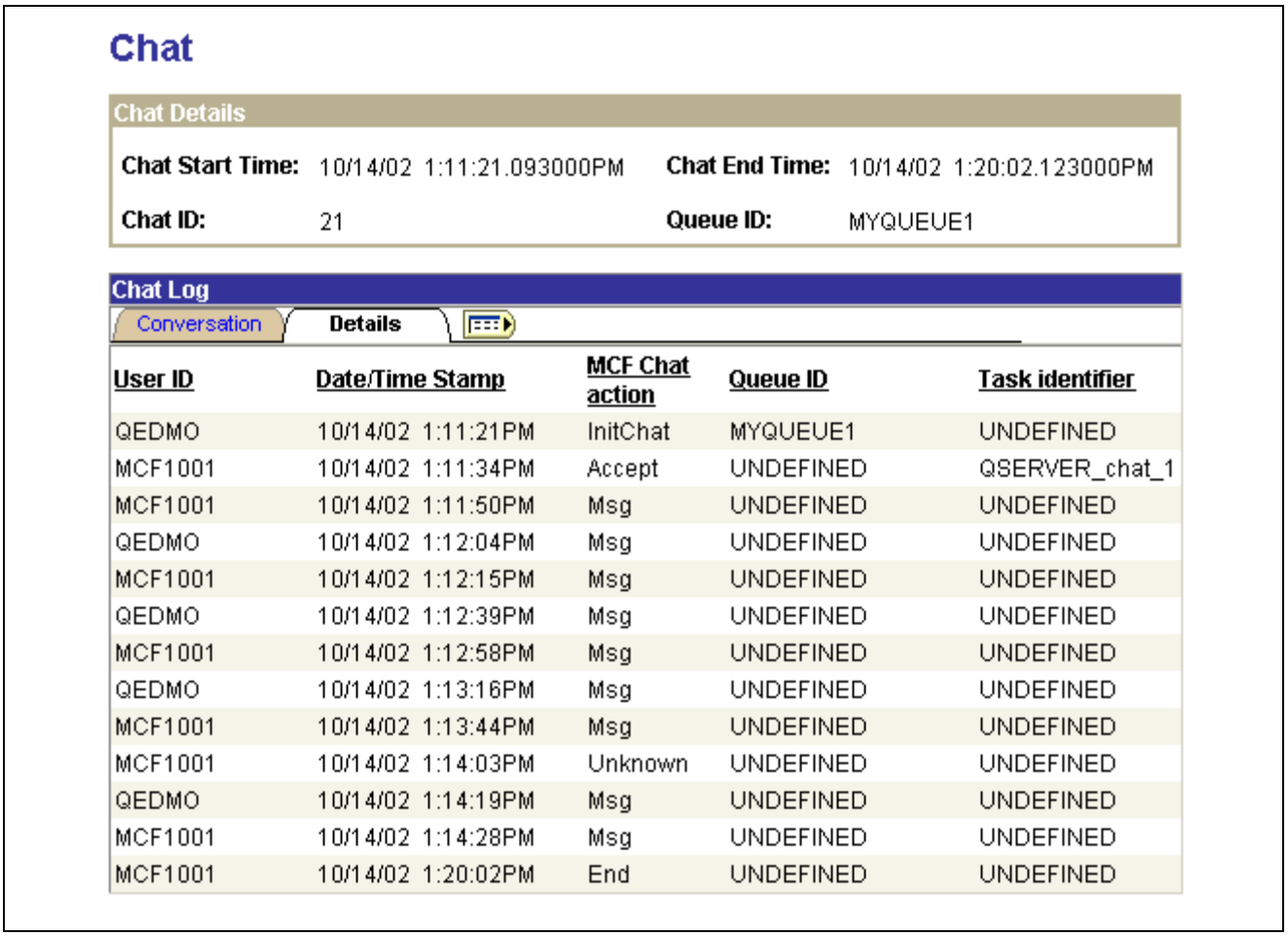
Chat page: Conversation tab

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

Select the Details tab.

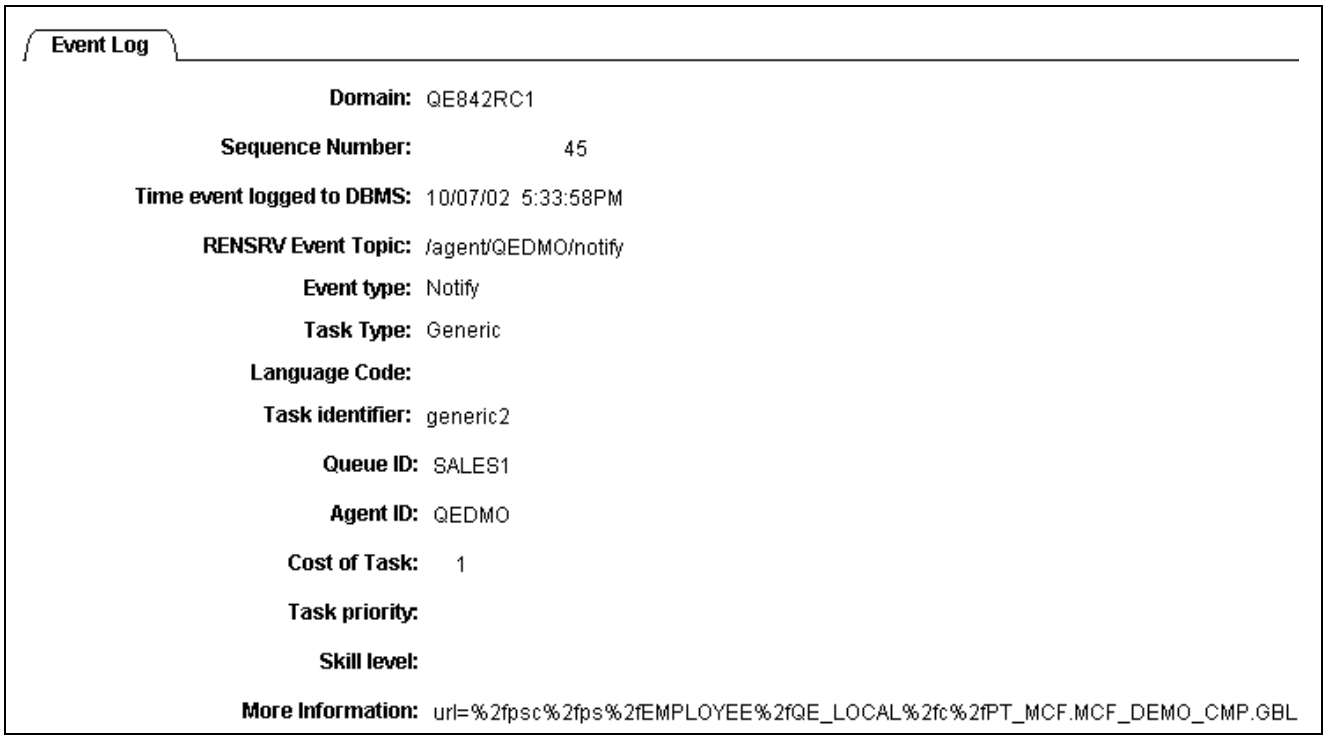


Chat page: Details tab

The Details tab displays detailed information about the selected chat conversation.

Viewing Event Logs

Access the Event Log page.



Event Log page

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:

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

The following table lists event log subtopics:

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 MCF Logs

Diagnostic PSUQSRV and PSMCFLOG traces are written to each application server domain's log directory. 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.

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.

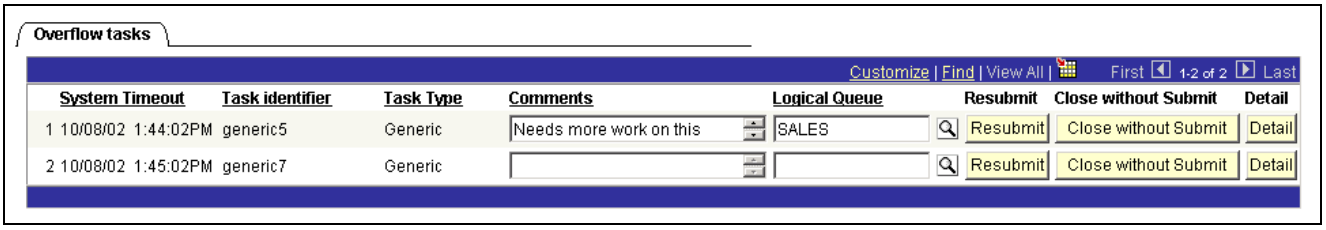
This section discusses how to administer overflow and escalated tasks.

Pages Used to Administer Overflow and Escalated Tasks

Page Name	Object 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.



Overflow tasks page

Use this page to manage tasks that could not be assigned to an agent within the specified overflow timeout.

System Timeout Displays the time at which the overflow occurred.

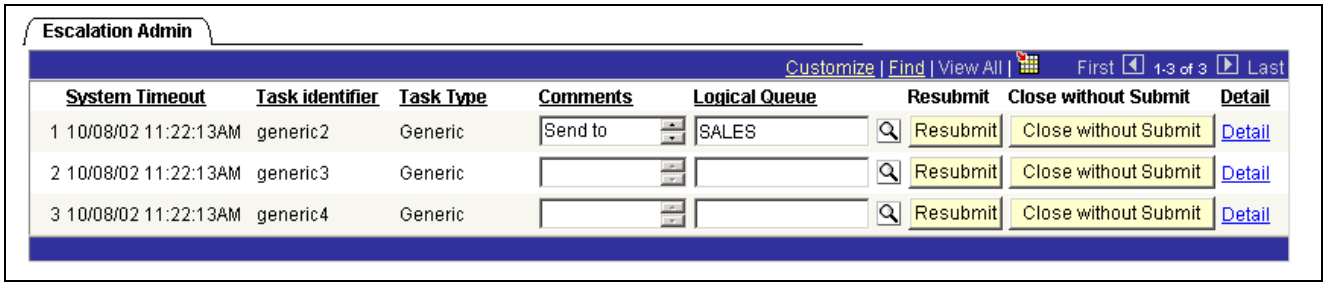
Task Type Types include chat, email, generic, and voice (not currently supported).

Comments Enter optional text commentary about the task's resolution.

	For example, note that the customer sent follow-up email that was answered by an agent.
Logical Queue	Select a logical queue to resubmit this task to.
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 Admin page.



Escalation Admin page

Use this page to manage accepted persistent tasks that have been automatically unassigned from agents because they were not closed within the specified escalation timeout. Tasks can be resubmitted or closed.

System Timeout	Displays when the escalation occurred.
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 specified logical queue to retry assignment. Only persistent tasks can be resubmitted; chat cannot be resubmitted.
Comments	Enter optional text commentary about the task's resolution. 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.

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.

MultiChannel Console

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.

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 task's application page may include a Done button; refer to the application's documentation.
- For chat, the task is done 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 appears.

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

The drop-down list box includes the physical queues that you can access.

To log on to a queue, select it.

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.

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:

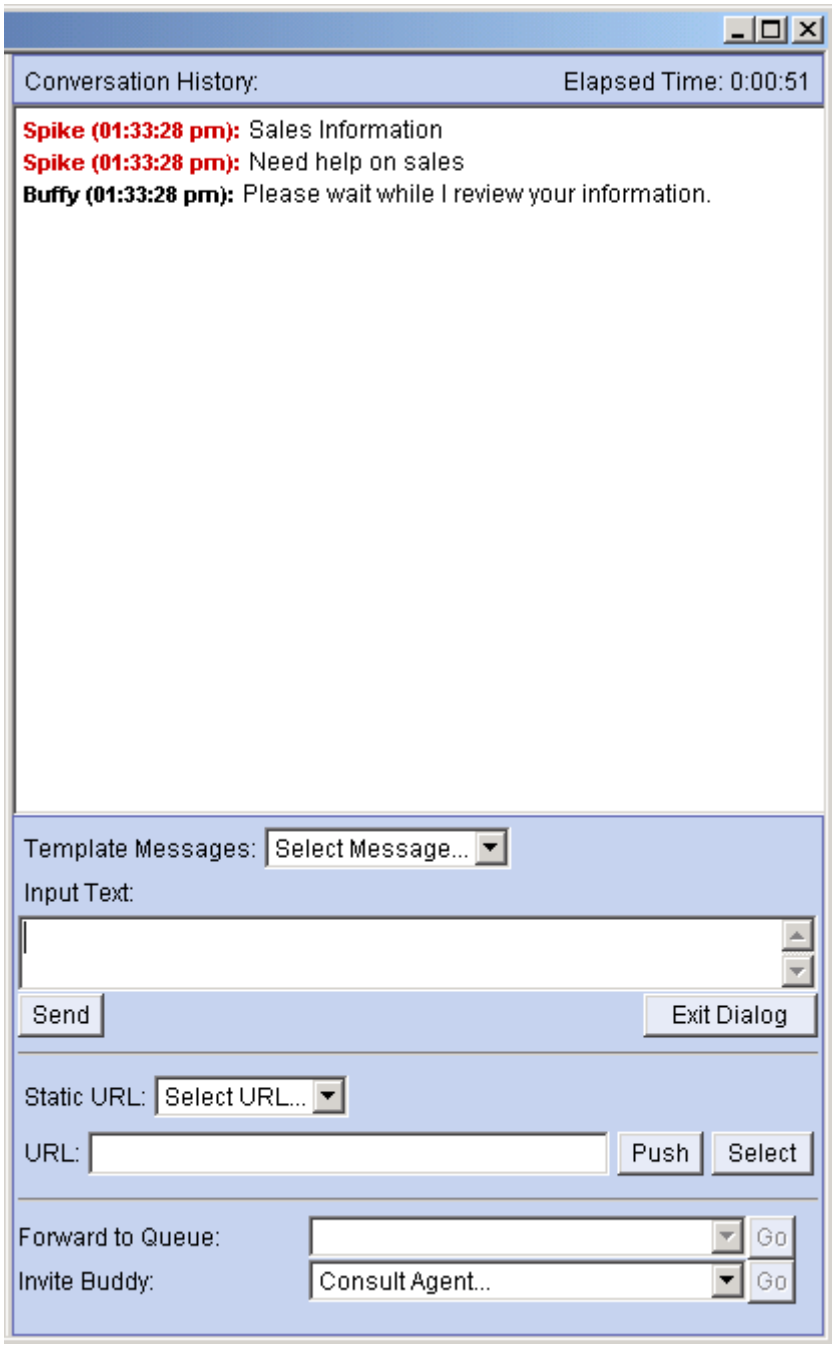
- Use the agent chat window.
- Use the customer chat window.

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.

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.

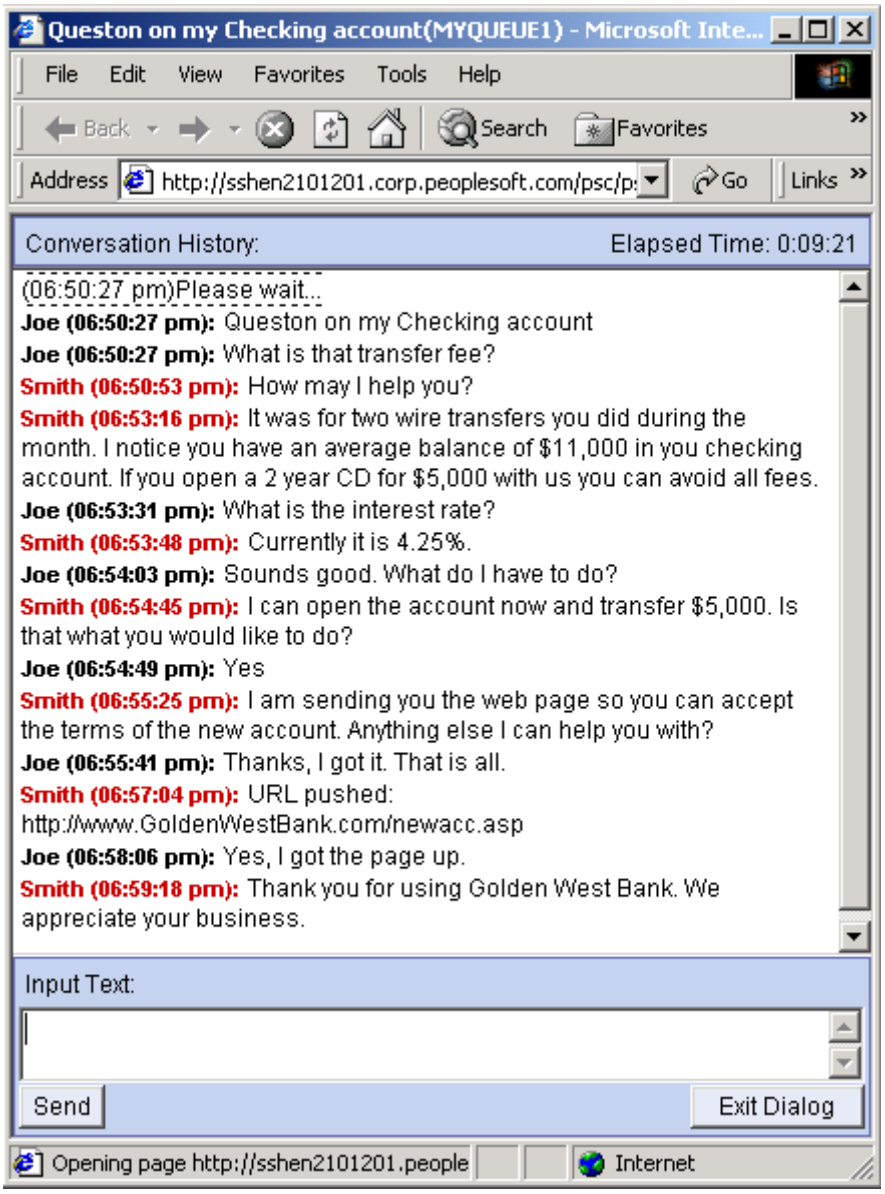
If chat logging is enabled, the conversation is recorded in the database chat log by the MCF log server. View the chat log on the Chat Log page.

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.
Template Messages	Send the customer a standard message by selecting one from the list. The message text appears in the Input Text text box. Click Send to send the message. Edit template messages for each queue on the queue Chat Responses page.
Input Text	To respond to the customer, enter text, then click Send or press ENTER. 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. 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. 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. 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. See Chapter 11, "Developing for PeopleSoft MultiChannel Framework," Using the URL Wizard, page 123.
Forward to Queue	To forward the current chat to another queue, select the queue. 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. 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:



Customer chat window

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.

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.

CHAPTER 11

Developing for PeopleSoft MultiChannel Framework

This chapter discusses how to:

- Work with sample pages.
- Use and demonstrate the JavaScript MultiChannel API (JSMCAPI).
- Use PeopleCode built-in functions.
- Use Universal Queue classes.

Note. Applications use the PeopleCode built-in function GenerateComponentContentRelURL() to build the relative URLs of the pages that appear when the agent accepts a task. This built-in function takes the portal node as a parameter. This node must have a content URI configured in the PeopleSoft Portal administration pages to ensure that the correct URL is generated.

Working with Sample Pages

To demonstrate MCF tools and functionality, use the MCF Sample Pages (MCF_DEMO_CMP) component.

This section discusses how to:

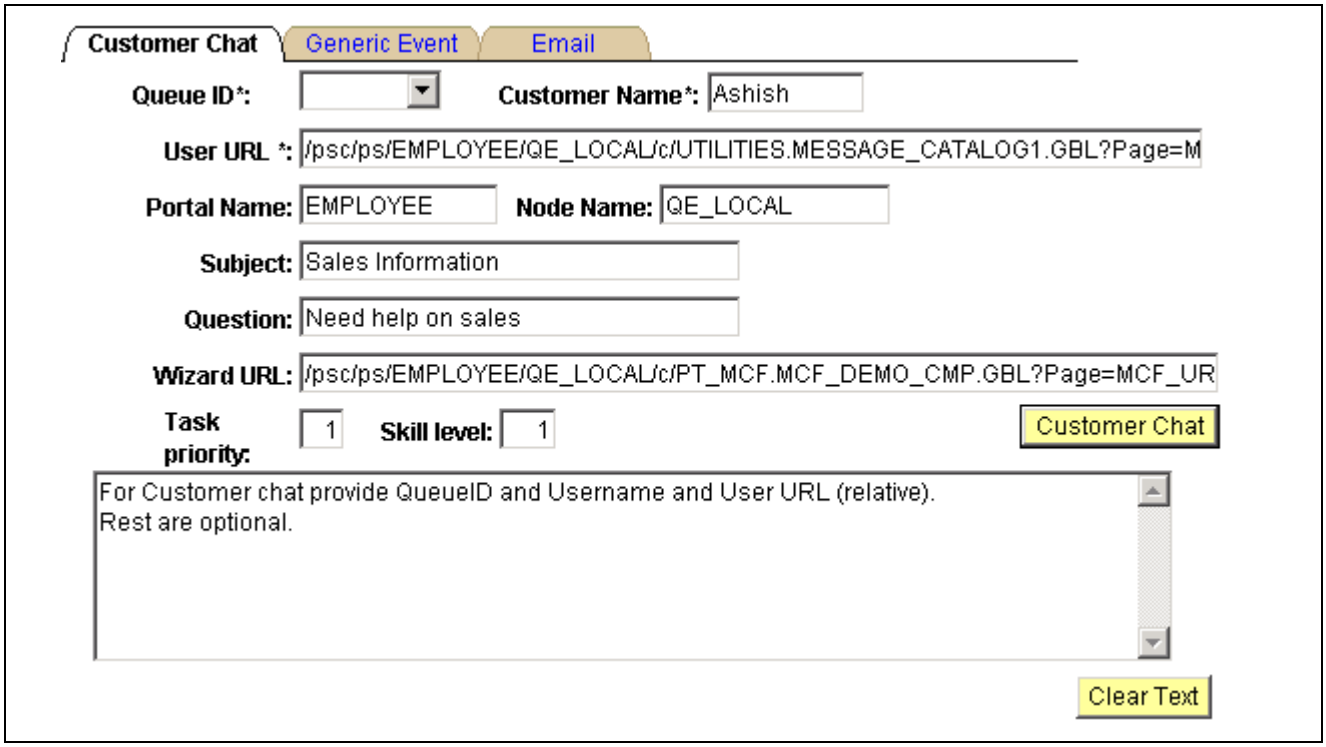
- Use the Customer Chat sample page.
- Use the URL wizard.
- Use the Generic Event sample page.
- Use the Generic Event window.
- Use the Email sample page.
- Use the Email window.

Pages Used to Work with Sample Pages

Page Name	Object 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.



Customer Chat page

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.

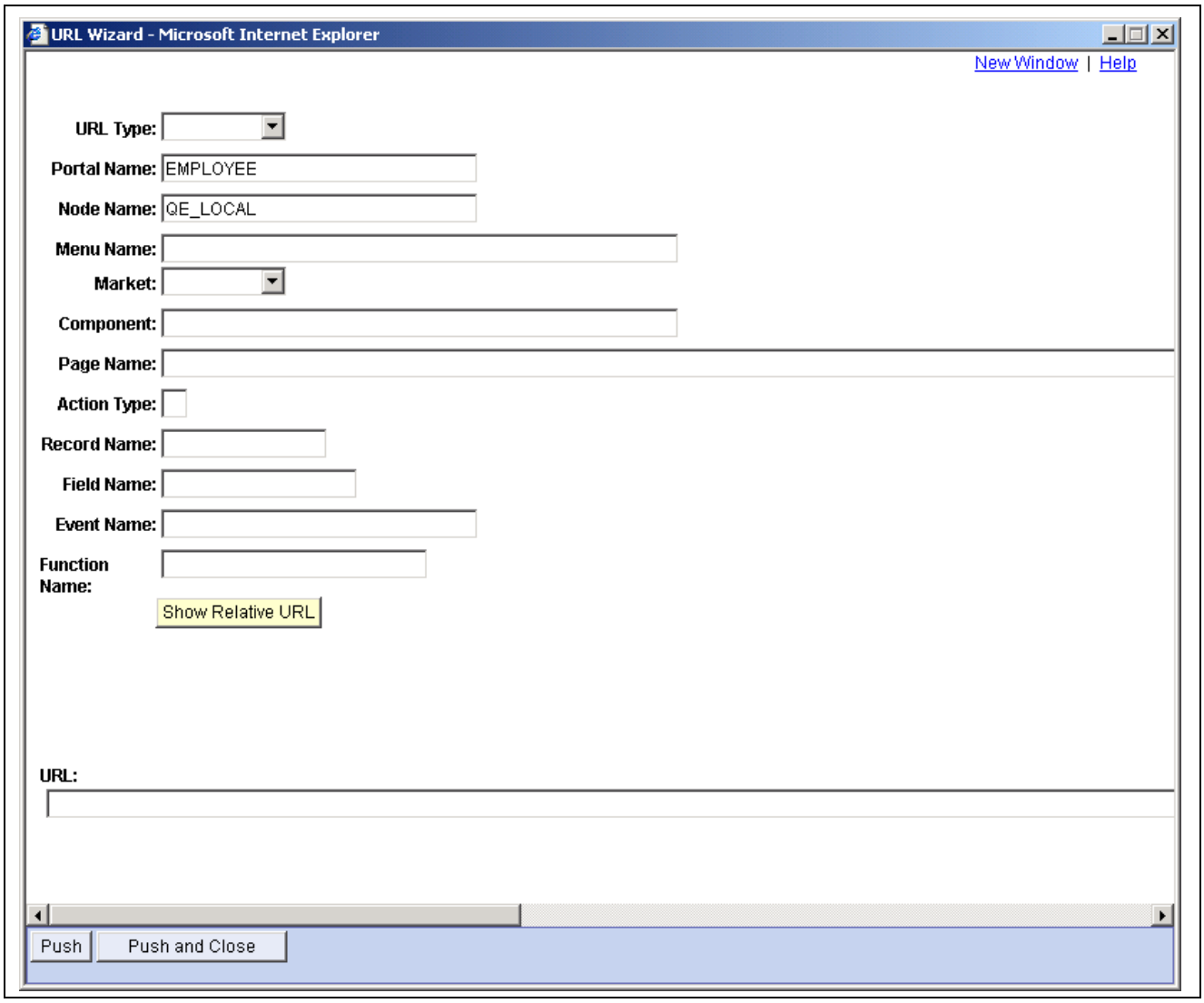
   The customer chat window appears.
5. On the MultiChannel Console, click the flashing chat notification icon to accept the chat session.

   The agent chat window appears.

   You can input 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:



URL Wizard page

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.

URL Type Select the type of URL being grabbed:

- *Component:* Generates a PeopleSoft Pure Internet Architecture component URL.

  Component URL generation uses the PeopleCode built-in function **GenerateComponentContentRelURL**.

  See *Enterprise PeopleTools 8.46 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 *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions".

Fields not required by the selected URL type are not available.

Show Relative URL Click to display the relative URL 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.

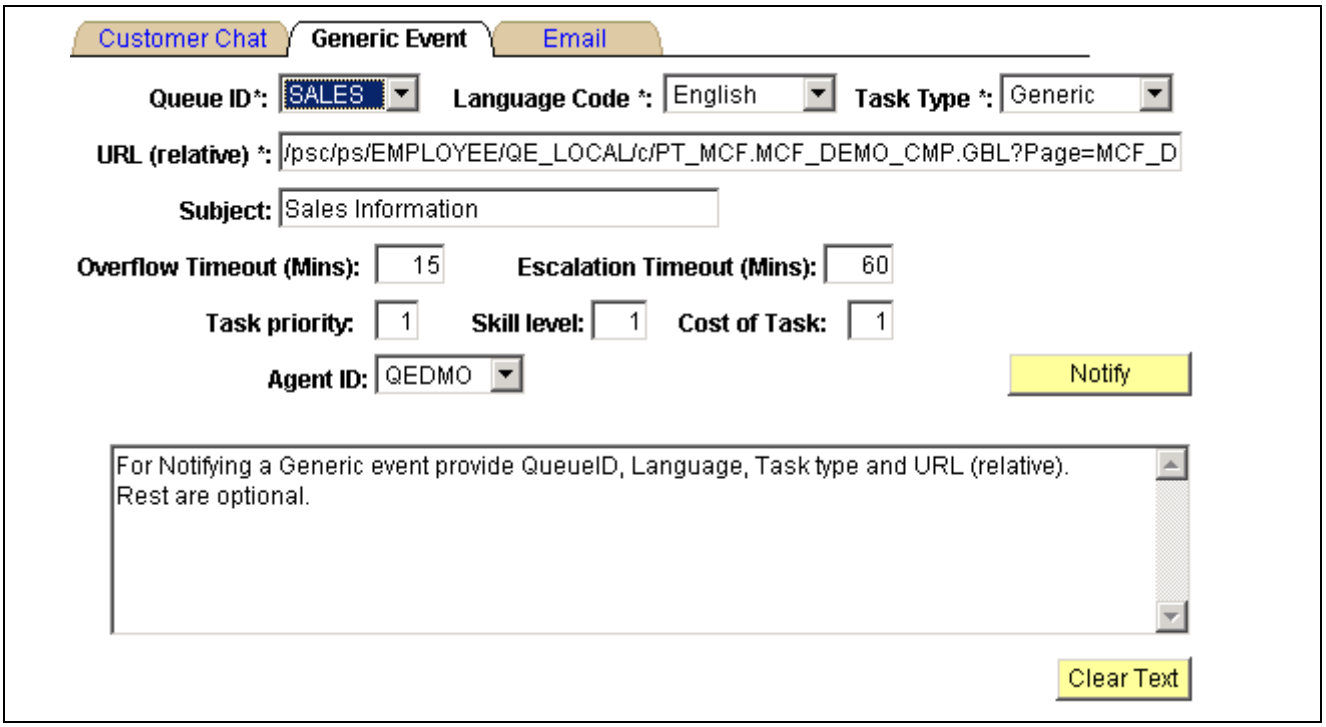
The Push button demonstrates functionality that must be included in an application-specific URL wizard.

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.



Generic Event page

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:

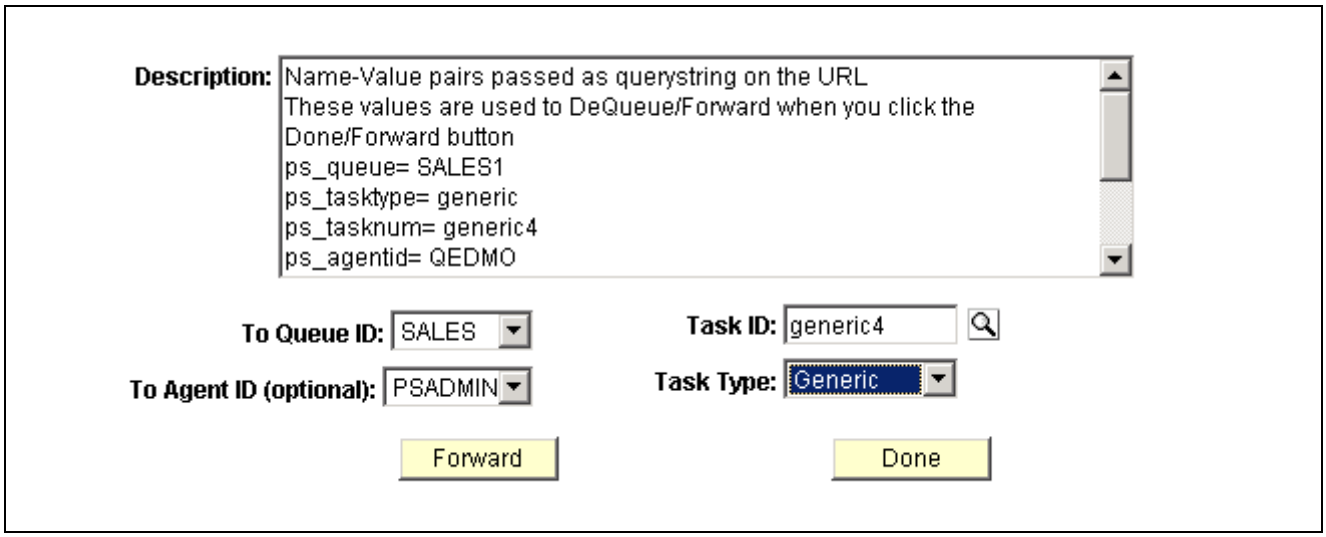
1. Access the MultiChannel Console.
2. Open the Generic Event page.
3. Open the Queue ID drop-down list box and select the queue that you are logged on to.
4. Open the Agent ID drop-down list box and select the agent ID that you are logged on with.
5. Select *Generic* for the task type.
6. Click Notify.
7. On the MultiChannel Console, click the flashing event notification icon to accept the event.

   The Generic Event window appears.

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:



Generic Event window

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.

The Forward 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.

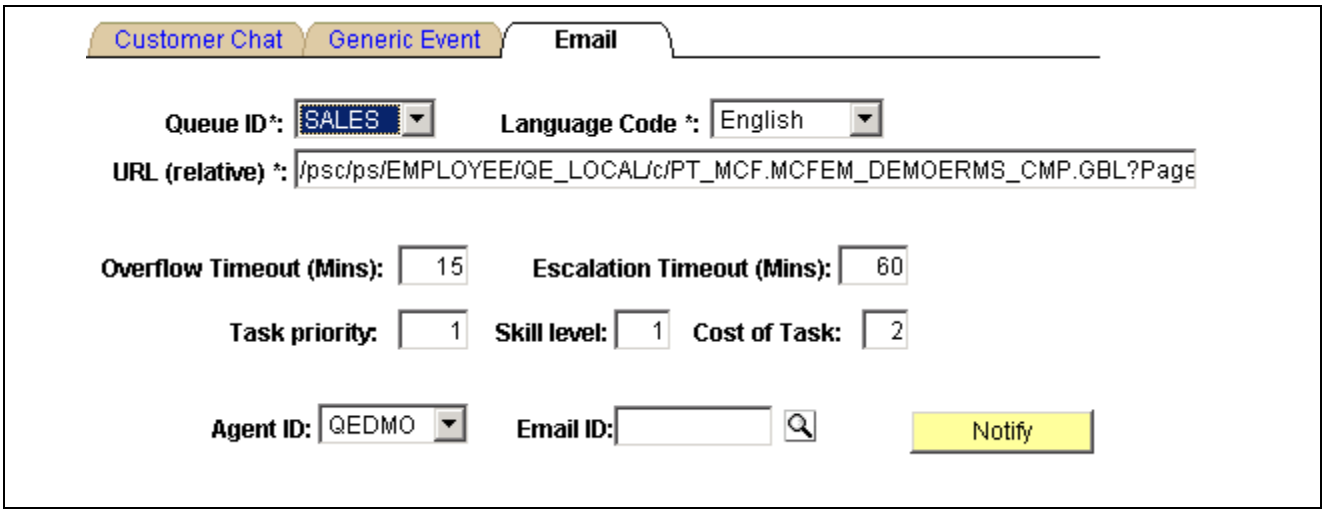
The Done button demonstrates the functionality of the DeQueue() built-in function.

See Also

Enterprise PeopleTools 8.46 PeopleBook: PeopleCode Language Reference, "PeopleCode Built-in Functions"

Using the Email Sample Page

Access the Email page.



Email page

The email sample pages are intended for demonstration purposes and should not be used in production.

EnQueue() and NotifyQ() may also be called from PeopleSoft Application Engine batch programs.

See *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode Language Reference*, "PeopleCode Built-in Functions".

Note. 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. Access the MultiChannel Console.
2. Open the Email 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. Click Notify.
6. On the MultiChannel Console, click the flashing email notification icon to accept the email.

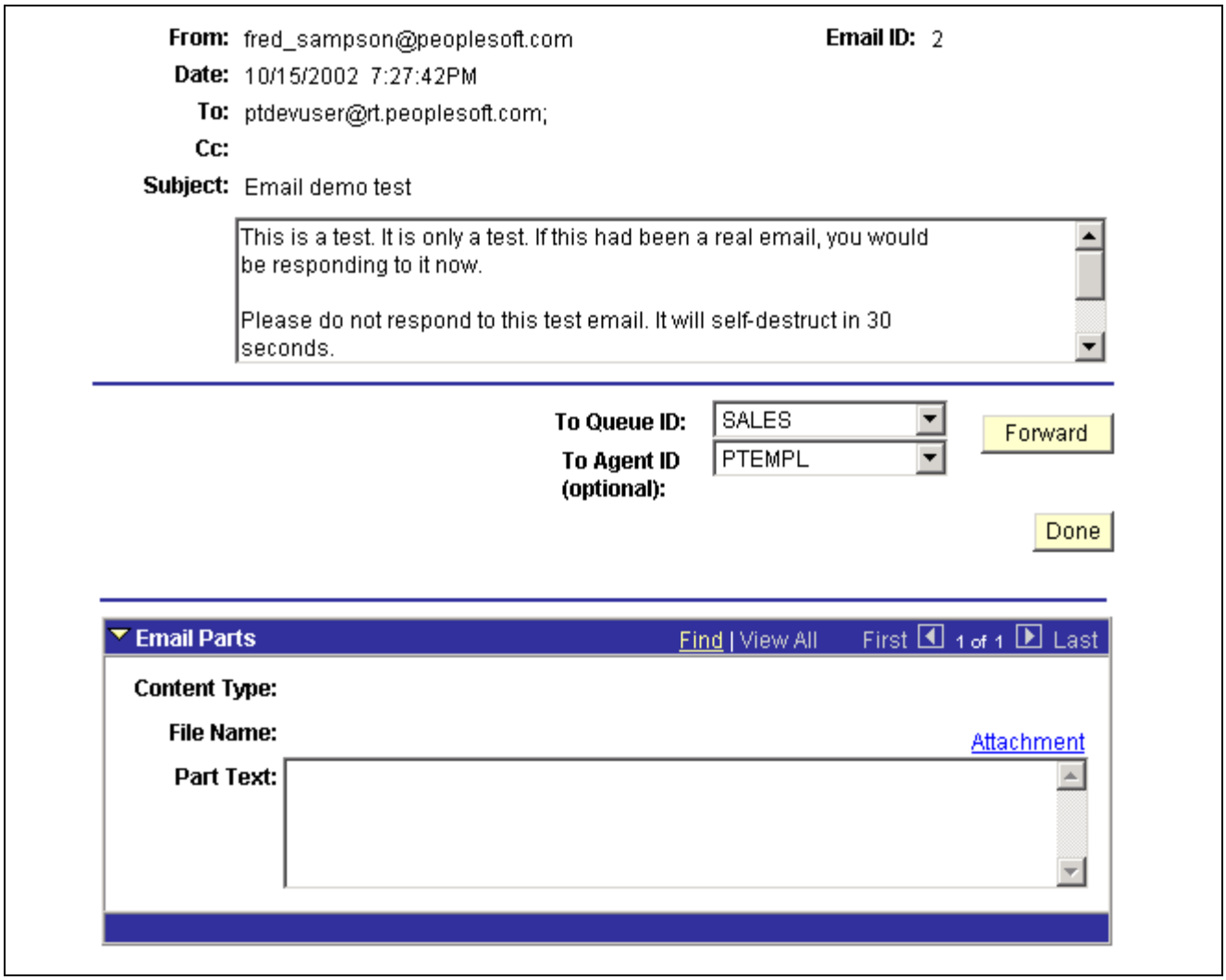
The email window appears.

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 agents Window Configuration page.

The following example shows an email window:



Email window

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.

Forward Click to forward the email to the specified queue or agent.

This Forward button demonstrates the functionality of the Forward() built-in function.

Done	Click to exit the email and remove it from the queue. The Done button demonstrates the functionality of the DeQueue() built-in.

Email Parts

If an email has been broken into parts stored in the email database, or has an attachment, they can be accessed here.

See Also

Chapter 13, "Configuring the Email Channel," Demonstrating the Email Channel, page 273

Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference, "Mail Classes"

Enterprise PeopleTools 8.46 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 JavaScript MultiChannel Application Programming Interface (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 real-time event notification (REN) JavaScript client. JSMCAPI uses standard JavaScript.

PeopleSoft provides sample pages demonstrating the functionality enabled by JSMCAPI. These pages are for demonstration purposes only, and should not be used in a production environment.

JSMCAPI is delivered with PeopleTools; there is no need for a separate installation. The jsmcapi.js file is located in the <PS_HOME>\webserv\<domain>\applications\peoplesoft\PORTAL\ps\pMCF folder.

See Also

Chapter 11, "Developing for PeopleSoft MultiChannel Framework," Pages Used to Demonstrate JSMCAPI, page 130

Chapter 12, "JSMCAPI Classes," page 145

Appendix A, "JSMCAPI Quick Reference," page 285

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 Specify 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 Represents three task types: generic, email, chat.

Click to initiate the action selected in the associated drop-down list box.

MCF Cluster ID Specify the MCF cluster to be monitored or on which to test the sample console.

Physical Queue Specify 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 The State field displays a value indicating the state of the associated element.

Statistics The Statistics fields display statistics generated by the CTI server.

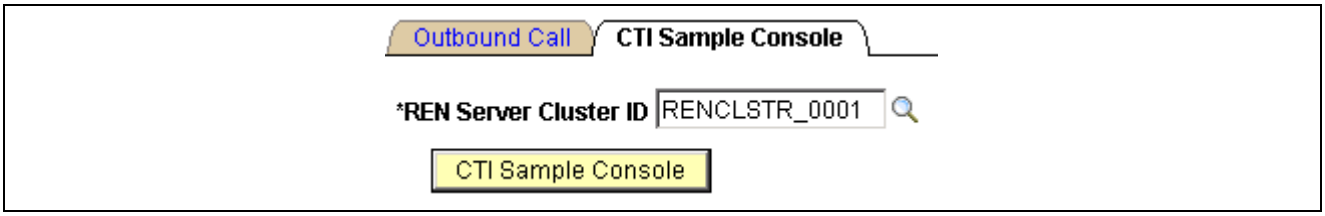
User Id The CTI agent's PeopleSoft user ID.

Pages Used to Demonstrate JSMCAPI

Page Name	Object Name	Navigation	Usage
Agent Console	PT_SAMPLEAGNTCONSL	PeopleTools, MultiChannel Framework, Universal Queue, Sample Pages, Agent Console	View a sample MultiChannel Console generated using JSMCAPI
CTI Sample Console	CTI_SAMPLECONSOLE	PeopleTools, MultiChannel Framework, CTI Configuration, Sample Pages, CTI Sample Console	View a sample CTI console 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.



CTI Sample Console page

REN Server Cluster ID Specify the REN server cluster on which to test the sample console.

CTI Sample Console Click to initiate the sample console.

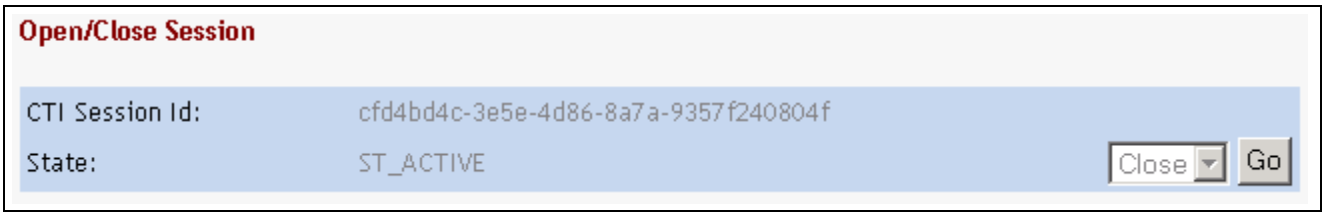
Note. To demonstrate the CTI sample console, you must have a CTI server running and communicating with the specified REN server cluster.

After clicking CTI Sample Console, a new browser window appears that displays the sample console.

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.

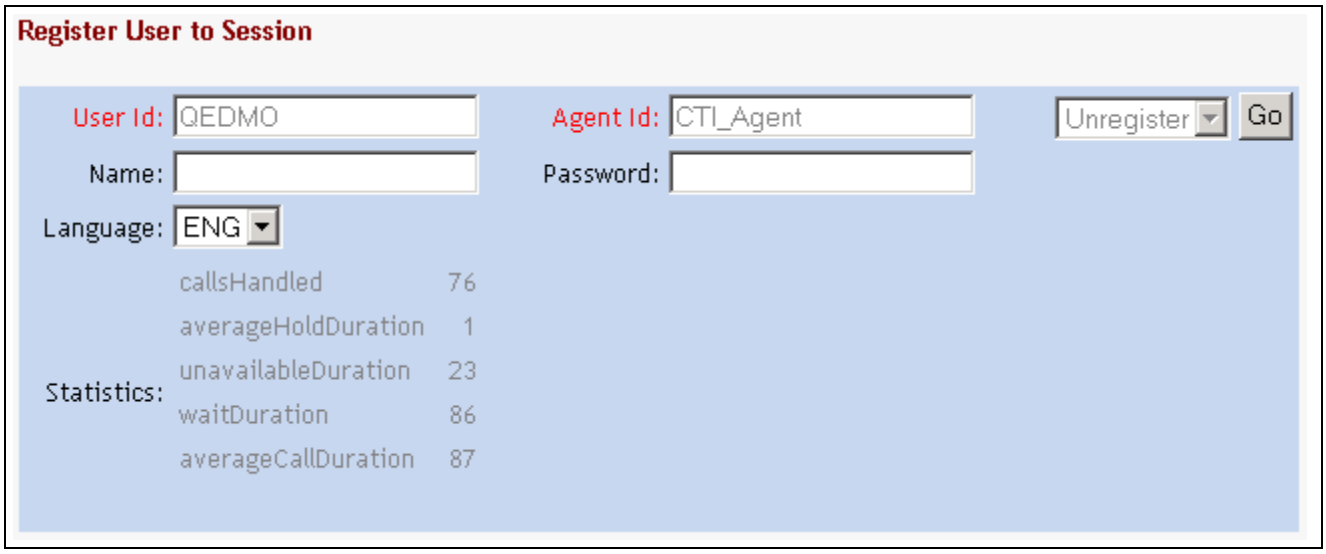
CTI Session Id The CTI Session Id field displays a value identifying the session.

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.



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.

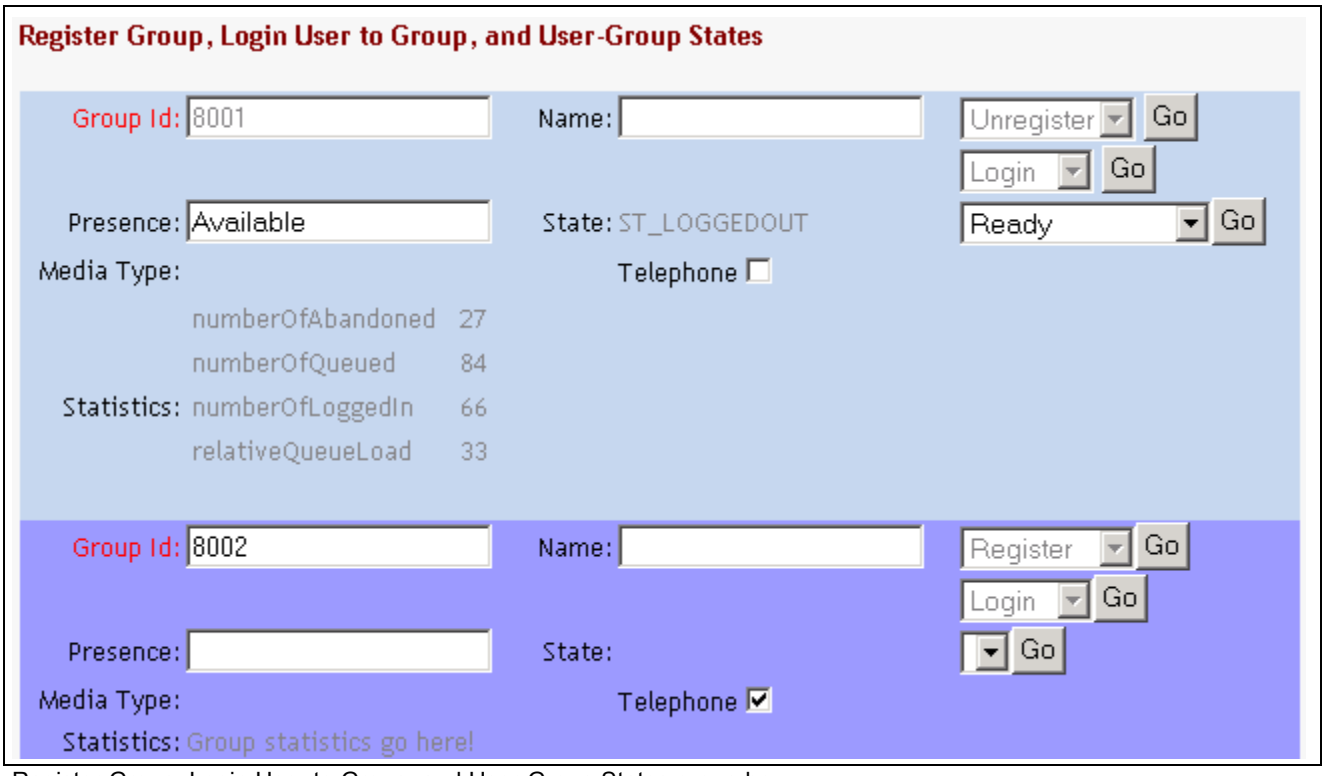
Language Reserved for future use.

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.



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.

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 pre-defined 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.

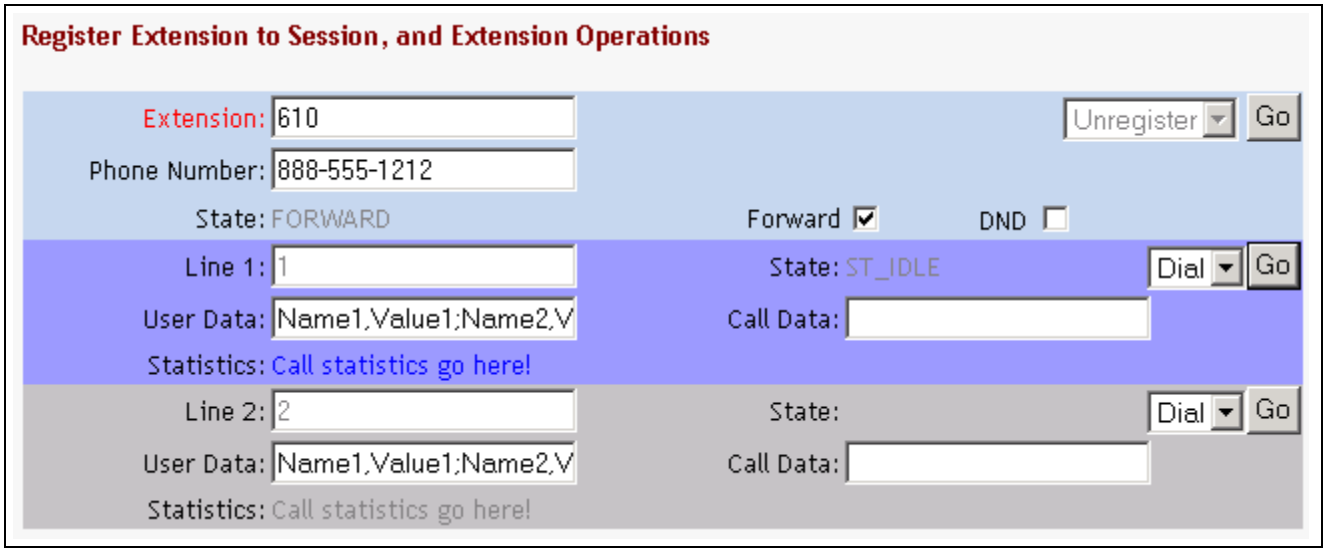
To enable incoming calls, select *Ready* from the drop-down list box and click the Go button.

Media Type Reserved for future use.

The sample console only supports 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.



Register Extension to Session, and Extension Operations group box

Extension Specify a valid telephone extension number.

Phone Number Specify a valid telephone number for use in subsequent operations, such as dial, forward, or do-not-disturb (DND).

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.

On the sample console, the value will either be *1* or *2*.

User Data Specify name-value pairs representing user data to be attached to a call.

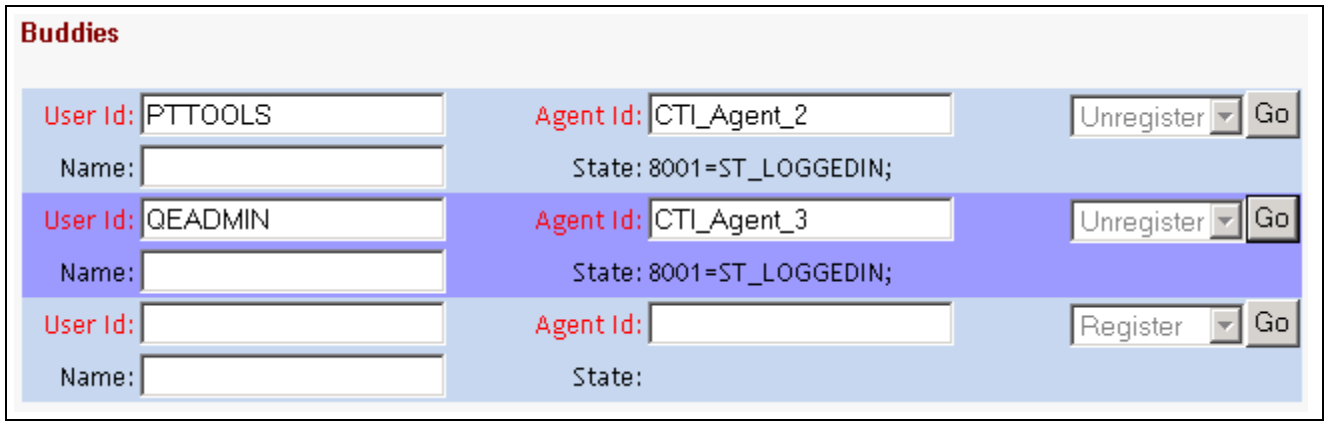
Call Data Specify any call data to be attached to the call.

Call data includes:

- *ANI*
- *DNIS*
- *THISDN*
- *OTHERDN*

Buddies

Access the Buddies group box.



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 different from the agent's user ID, both values are required.

Server State and Broadcasts

Access the Server State and Broadcasts group box.

Server State and Broadcasts group box

CTI State Displays the status of the CTI server.

REN State Displays the status of the REN server.

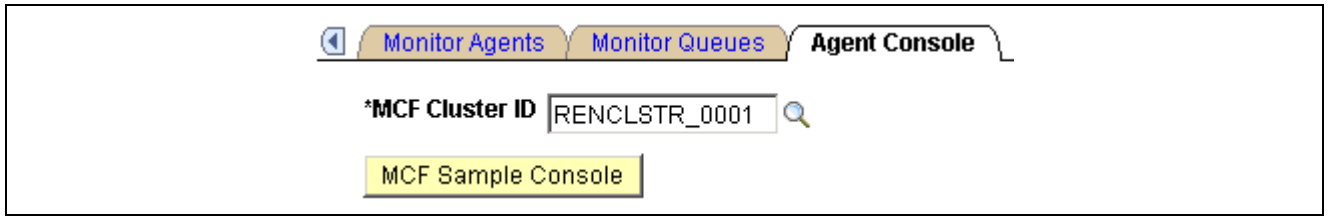
Info (information) Displays the text of a broadcast message from the CTI server.

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.



Agent Console page

MCF Sample Console Click to initiate the sample console.

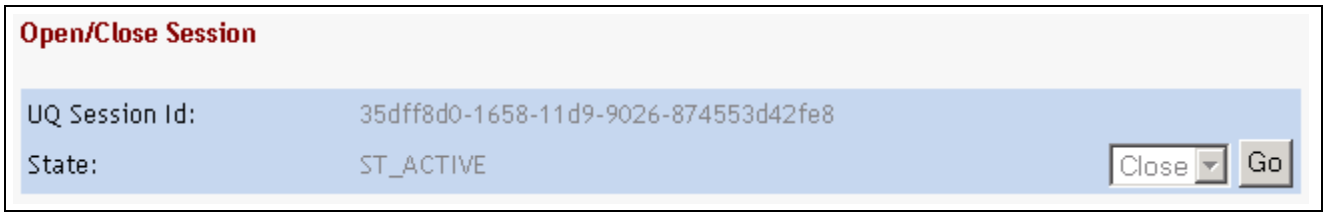
Note. To demonstrate the MCF sample console, you must have an MCF cluster running and communicating with the specified REN server cluster.

After clicking MCF Sample Console, a new browser window appears that displays the sample console.

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.

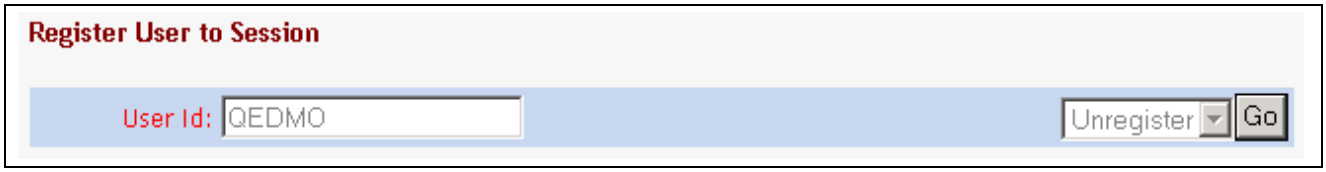


Open/Close Session group box

UQ Session Id The universal queue session ID.

Register User to Session

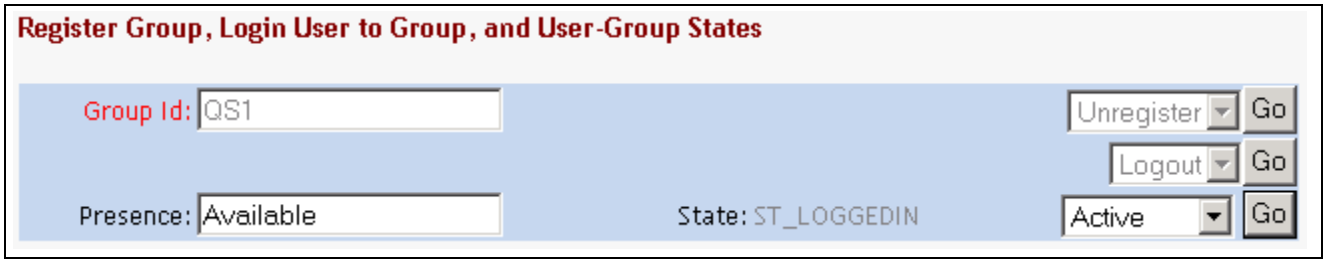
Access the Register User to Session group box.



Register User to Session group box

Register Group, Login User to Group, and User-Group States

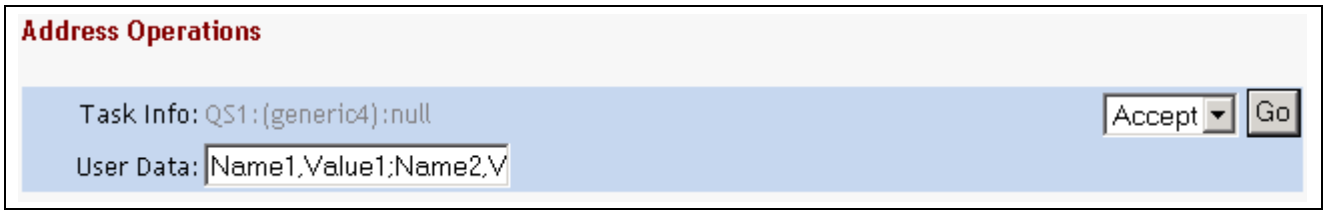
Access the Register Group, Login User to Group, and User-Group States group box.



Register Group, Login User to Group, and User-Group States group box

Address Operations

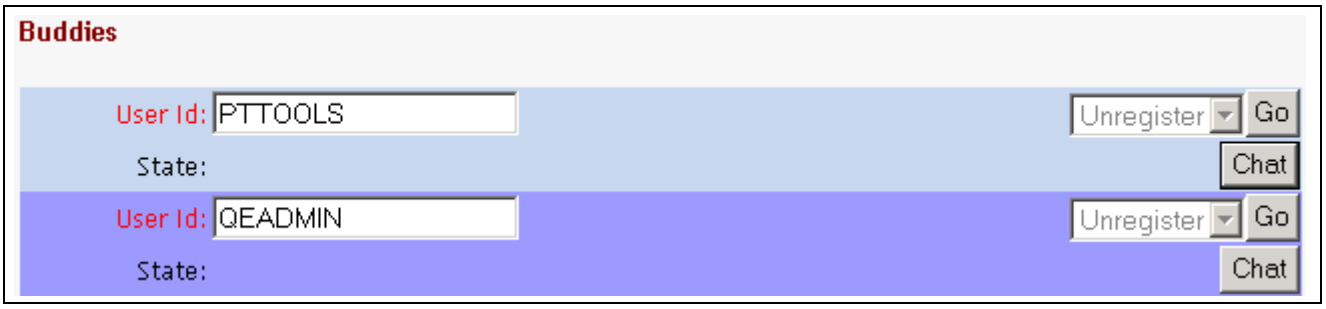
Access the Address Operations group box.



Address Operations group box

Buddies

Access the Buddies group box.

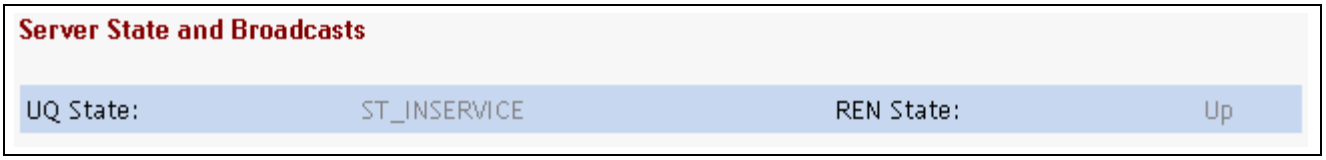


Buddies group box

Click to initiate an agent-to-agent chat session.

Server State and Broadcasts

Access the Server State and Broadcasts group box.



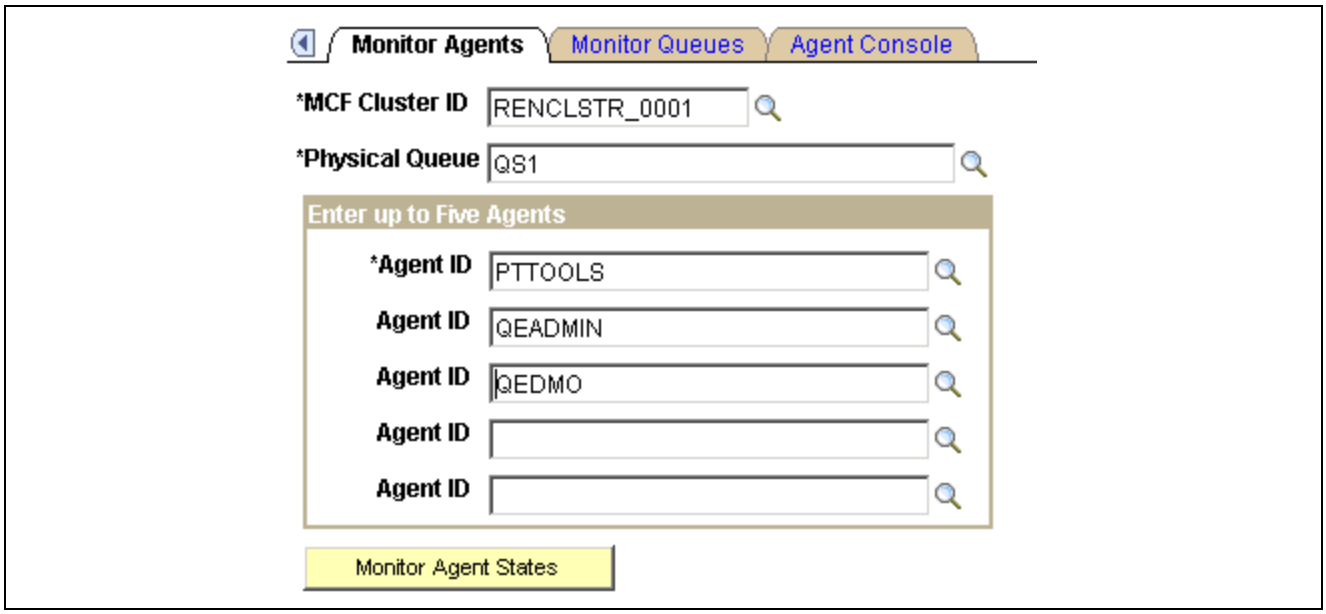
Server State and Broadcasts group box

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.

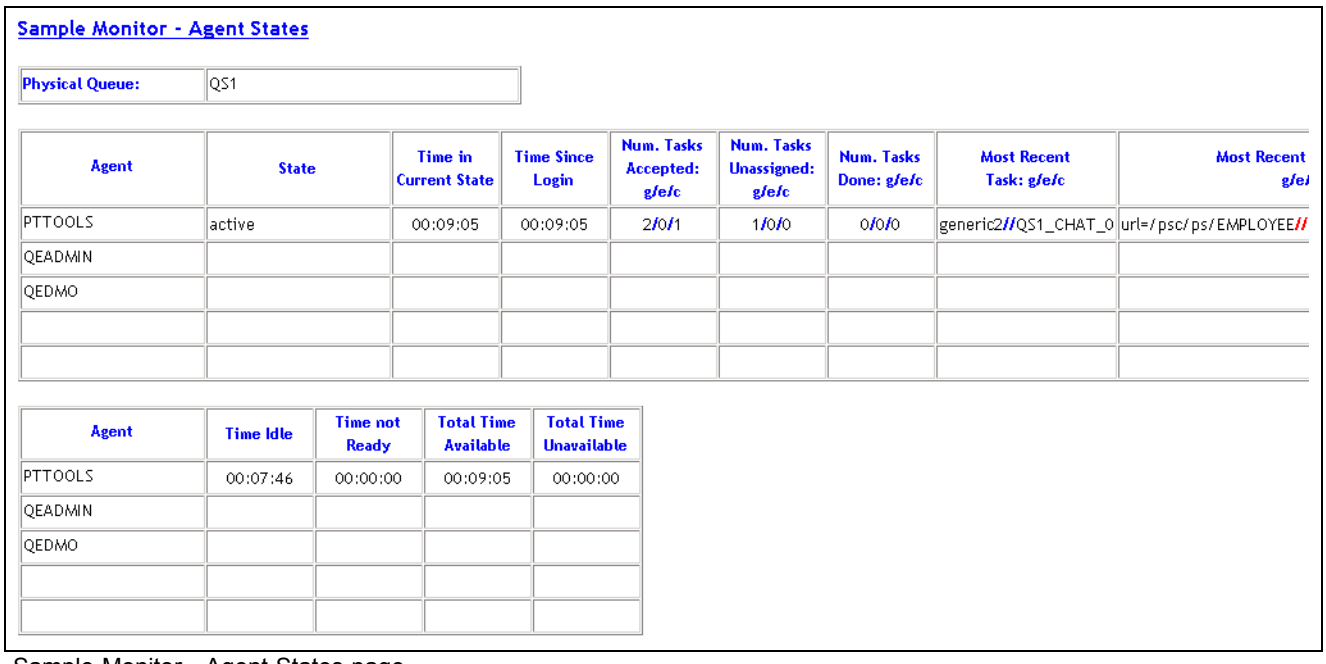


Monitor Agents page

Specify the MCF cluster, physical queue on that cluster, and up to five agents on that queue to be monitored.

After clicking Monitor Agent States, a new browser window appears.

Note. To enable the new browser window to appear, disable any pop-up blocking software for your browser.



Sample Monitor - Agent States page

Physical Queue

The ID of the physical queue being monitored appears in the first table.

Agent Statistics Table 1

The first agent statistics table displays information published by the JSMCAPI user.onStat1 event..

Time in Current State	Displays the duration of the agent's current state. Returned by `timeInCurrentState`.
Time Since Login	Denotes the duration since the agent logged in. Returned by `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. Returned by `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`.
Most Recent Task: g/e/c	Identifies the most recent task accepted since login for each task type. Returned by `mostRecentTaskId`.
Most Recent Task Data: g/e/c	Displays data on the most recent task accepted since login for each task type. Returned by `mostRecentTaskInfo`

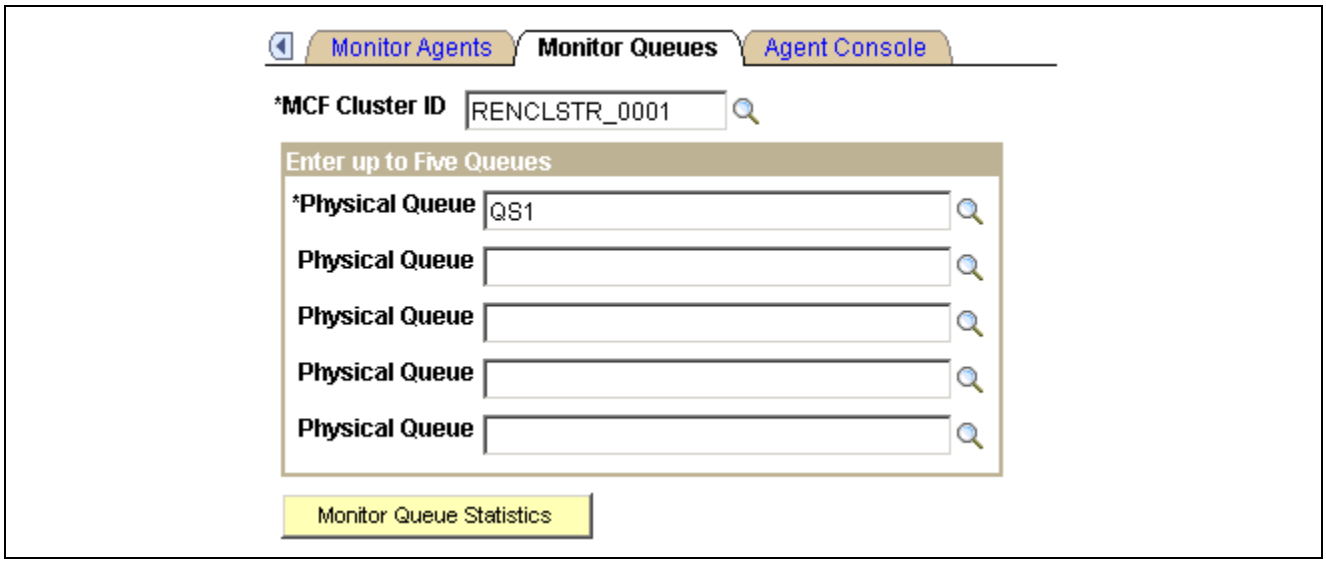
Agent Statistics Table 2

The second agent statistics table displays information 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. Returned by `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. Returned by `timeNotReady`.
Total Time Available	Displays the total amount of time the agent has been in an available status. Returned by `totalTimeAvailable`.
Total Time Unavailable	Displays the total amount of time the agent has been in an unavailable status. Returned by `totalTimeUnavailable`.

Using the Monitor Queues Page and Sample Monitor - Queue Statistics Page

Access the Monitor Queues Page.

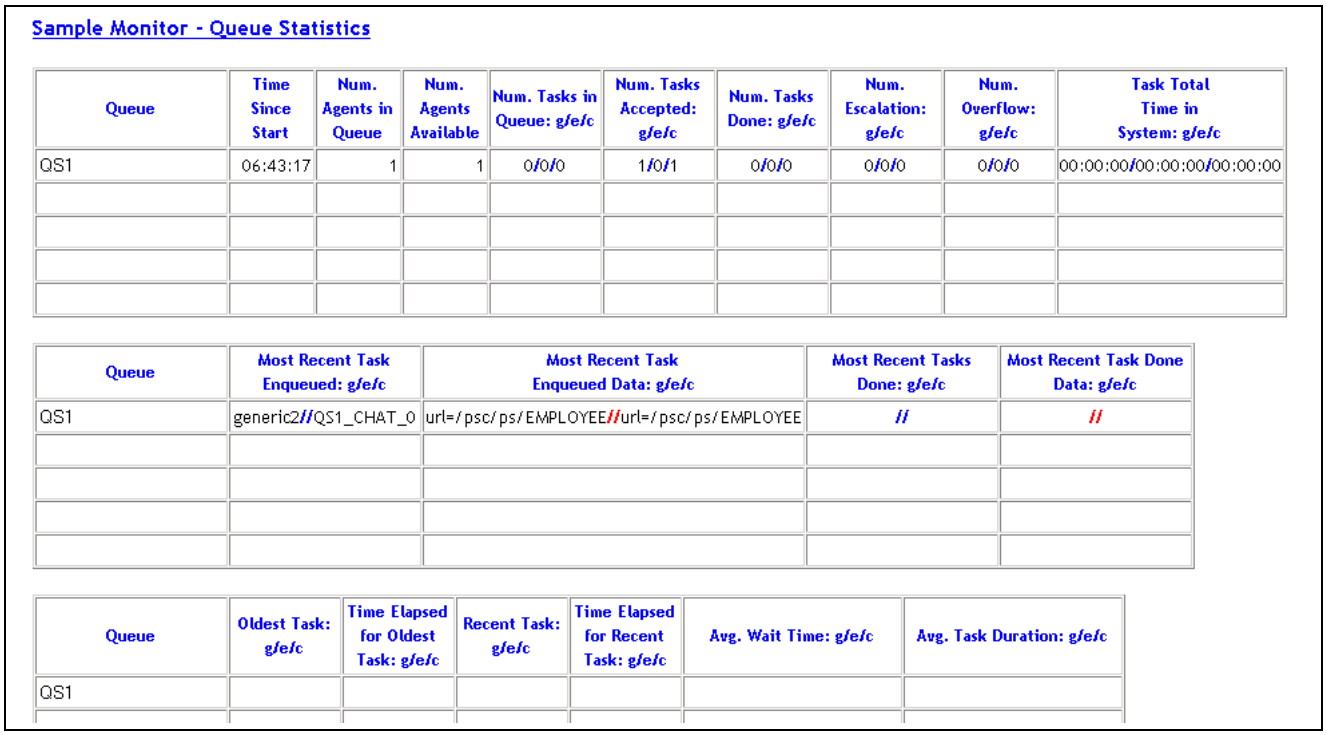


Monitor Queues page

Specify the MCF cluster and up to five physical queues on that cluster to be monitored.

After clicking 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.



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 published by the JSMCAPI group.onStat1 event.

Queue	Displays an identifier for this queue. Returned by `Queue_Id`.
Time Since Start	The duration since this queue was started (application server domain start). Returned by `timeSinceStart`.
Num. Agents in Queue (number of agents in queue)	The number of agents currently logged in to this queue. Returned by `numAgentsLoggedIn`.
Num. Agents Available (number of agents available)	The number of agents currently logged in to this queue with an available status. Returned by `numAgentsAvail`.
Num. Tasks in Queue: g/e/c (number of tasks in queue)	The number of tasks currently on this queue, by task type. Returned by `numTasksQueued`.
Num. Tasks Accepted: g/e/c (number of tasks accepted)	The number of accepted tasks currently on this queue, by task type. Returned by `numTaskAccepted`.
Num. Tasks Done: g/e/c (number of tasks done)	The number of tasks on this queue that have been completed, by task type. Returned by `numTaskDone`.
Num. Escalation: g/e/c (number of escalation)	The number of escalated tasks on this queue, by task type. Returned by `numEscalation`.
Num. Overflow: g/e/c (number of overflow)	The number of overflow tasks on this queue, by task type. Returned by `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 the task was enqueued and the time at which it was marked done. Returned by `taskTotalTimeInSystem`.

Queue Statistics 2

The second queue statistics table displays data that changes frequently. The table displays information published by the JSMCAPI group.onStat1 event.

Most Recent Task Enqueued: g/e/c	Identifies the task most recently enqueued on this queue, by task type. Returned by `mostRecentTaskEnqueued`.
Most Recent Task Enqueued Data: g/e/c	Displays data for the most recently enqueued task on this queue, by task type. Returned by `mostRecentTasksEnqueuedData`.
Most Recent Tasks Done: g/e/c	Identifies the task most recently completed, by task type. Returned by `mostRecentTaskDone`.
Most Recent Task Done Data: g/e/c	Displays data for the most recently completed tasks, by task type. Returned by `mostRecentTaskDoneData`.

Queue Statistics 3

The table displays information 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.

Returned by `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.

Returned by `timeElapsedOldestTask`.

Recent Task: g/e/c

Identifies the most recent task enqueued and accepted on this queue, by task type.

Returned by `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.

Returned by `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.

Returned by `averageWaitTime`.

Avg. Task Duration: g/e/c (average task duration)

Displays the average duration before a task is completed.

Returned by `averageTaskDuration`.

The average task duration is calculated by referencing a list of completed tasks. The size of that list is determined by the donelistsize parameter set on the Cluster Tuning page.

See Chapter 7, "Configuring MCF Queues," Tuning Cluster Parameters, page 82.

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 function 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.

- NotifyQ

  Use NotifyQ to notify the queue server of a task enqueued by EnQueue.

See *Enterprise PeopleTools 8.46 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 *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference*, "Universal Queue Classes".

CHAPTER 12

JSMCAPI Classes

This chapter discusses the JavaScript MultiChannel Application Programming Interface (JSMCAPI) classes.

Understanding JSMCAPI

This section introduces the JSMCAPI, a JavaScript-based 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 upon the REN server JavaScript client and is a pure JavaScript API.

JSMCAPI is the interface with which the application developer accessess the JSMCAPI functionalities. 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.

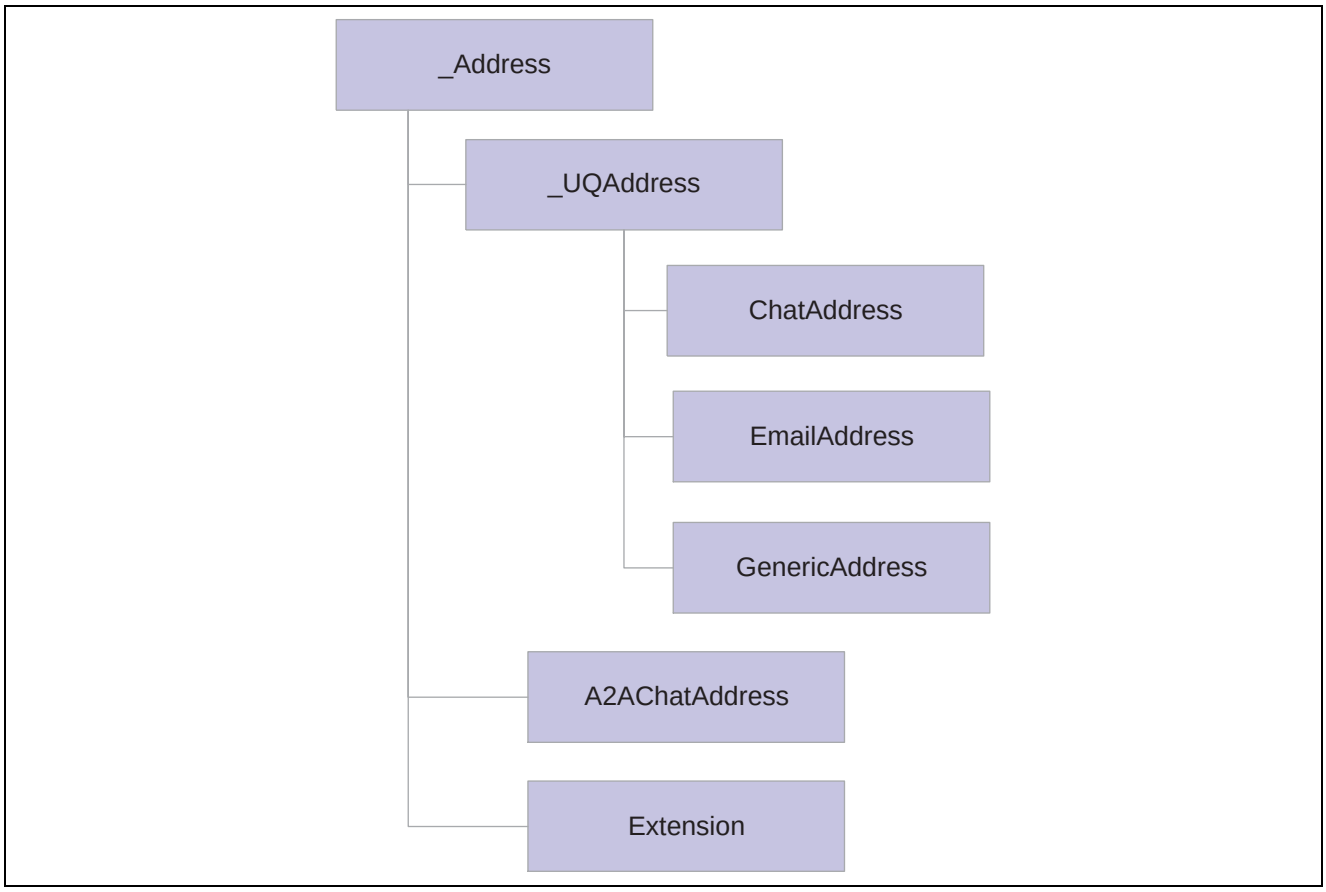
- Receives PBX/Switch events from the CTI server.

  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.

_Address Class Hierarchy

The _Address class can be extended by other subclasses. The following flowchart shows the different subclasses, and how they interrelate.

Do not instantiate _Address or _UQAddress class objects. Instead, use the child classes.



_Address class hierarchy

See Also

Chapter 12, “JSMCAPI Classes,” _Address Class Constructor, page 146

Chapter 12, “JSMCAPI Classes,” _UQAddress Class Constructor, page 148

Chapter 12, “JSMCAPI Classes,” A2AChatAddress Class Constructor, page 157

Chapter 12, “JSMCAPI Classes,” Extension Class Constructor, page 177

Chapter 12, “JSMCAPI Classes,” ChatAddress Class Constructor, page 172

Chapter 12, “JSMCAPI Classes,” EmailAddress Class Constructor, page 176

Chapter 12, “JSMCAPI Classes,” GenericAddress Class Constructor, page 186

_Address Class Constructor

The following is the _Address class constructor.

_Address

Syntax

```
_Address()
```

Description

Creates an _Address object that describes the address.

Parameters

None.

Returns

An _Address object.

_Address Class Fields

In this section we discuss the JSMCAPI _Address class fields. These fields are described in alphabetical order.

caps

Description

The capacities of the address.

Type: object.

id

Description

The ID of the address.

Type: string.

_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.

_UQAddress Class Fields

The _UQAddress class inherits the following fields from the _Address class.

- caps
- id

See Chapter 12, “JSMCAPI Classes,” _Address Class Fields, page 147.

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.

Returns

None.

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

None.

_UQAddress Class Callback Event Methods

The _UQAddress class inherits the onError callback event method from the _Address class.

See Chapter 12, "JSMCAPI Classes," _Address Class Callback Event Method, page 147.

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*)

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*)

Description

Fires when the task is added to the addressed queue.

onTaskRemoved

Syntax

onTaskRemoved(*event*)

Description

Fired when the task is removed

onUnassigned

Syntax

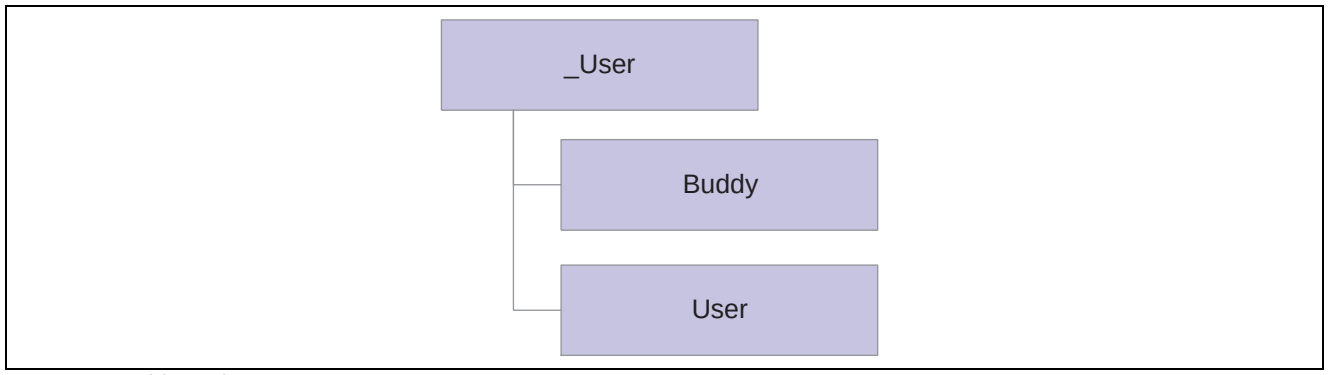
onUnassigned(*event*)

Description

Fired when the task is unassigned.

_User Class Hierarchy

The _User class can be extended by other subclasses. The following flowchart shows the different subclasses, and how they interrelate.



_User class hierarchy

See Also

Chapter 12, "JSMCAPI Classes," Buddy Class Constructor, page 164

Chapter 12, "JSMCAPI Classes," User Class Constructor, page 242

_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.

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_*.

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.

ST_WORKREADY

Description

The agent is in the work ready state.

statistics

Description

Agent statistics for CTI.

Type: AgentStatistics object.

statistics1

Description

Agent statistics for the universal queue server.

Type: UserStatistics1 object.

statistics2

Description

Agent statistics for the universal queue server.

Type: UserStatistcs2 object.

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	Specify the event associated with the agent-to-agent chat.
address	Specify the address from A2AChatAddress.
chatType	Specify the chat type.

Returns

An agent-to-agent chat object.

A2AChat Class Fields

The following are the A2AChat fields.

address

Description

The address containing the agent-to-agent chat.

Type: A2AChatAddress object.

See Also

Chapter 12, "JSMCAPI Classes," A2AChatAddress Class Constructor, page 157

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 12, "JSMCAPI Classes," AppData Class Constructor, page 161

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 12, "JSMCAPI Classes," GLOBALS Class Fields, page 190

id

Description

The ID of this chat.

Type: string with PS_JR constant.

jr

Description

JournalRoute

Type: string.

type

Description

The task type, chat.

Type: string.

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.

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 12, "JSMCAPI Classes," _Address Class Fields, page 147.

The following are the fields associated with the A2AChatAddress class.

id

Description

The address ID.

Type: string.

tasks

Description

The associative array of A2AChats.

Type: associative array.

See Also

Chapter 12, "JSMCAPI Classes," GLOBALS Class Fields, page 190

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 you wish to chat with.

Returns

None.

A2AChatAddress Class Callback Event Methods

The A2AChatAddress class inherits the onError callback event method from the _Address class.

See Chapter 12, "JSMCAPI Classes," _Address Class Callback Event Method, page 147.

The following are the A2AChatAddress class callback methods:

onChatEnded

Syntax

onChatEnded(*event*)

Description

Fires when an agent-to-agent chat is closed.

onInitiatingChat

Syntax

onInitiatingChat(*event*)

Description

Fires when initiating a chat.

onNotify

Syntax

onNotify(*event*)

Description

Fires when receiving the event notification.

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: string.

averageHoldDuration

Description

The average hold duration, in seconds, for an agent.

Type: string.

callsHandled

Description

The total number of calls handled by an agent.

Type: string.

data

Description

An associative array of key value pairs that includes all agent statistics.

Type: associative array.

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.

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.

jr

Description

The JournalRoute.

Type: string.

question

Description

The question posed by the chat initiator.

Type: string.

strData

Description

The raw application data.

Type: string.

subject

Description

The subject of the chat.

Type: string.

uniqueId

Description

The unique ID.

Type: string.

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 chat wizard URL.

Type: string.

AppData Class Method

The following is the AppData class method.

addKeyValue

Syntax

```
addKeyValue(kdy, 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 12, "JSMCAPI Classes," _User Class Constructor, page 151.

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.

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 12, "JSMCAPI Classes," _User Class Constructor, page 151.

Buddy Class Callback Event Methods

The following are the Buddy class callback event methods.

onStat1

Syntax

onStat1(*event*)

Description

Fires when statistics1 is received.

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 12, "JSMCAPI Classes," Task Class Constructor, page 240.

The following is the Call class constructor.

Call

Syntax

Call(*strCall*)

Description

The Call object describes the call task information associated with the line.

Parameters

Parameter	Description
strCall	Specify the call to use with this object.

Returns

A Call object.

Call Class Fields

The Call class inherits the following fields from Task:

- caseid
- cost
- customerid
- group
- id
- onStat
- priority
- type
- urlAbs
- urlRel

See Chapter 12, "JSMCAPI Classes," Task Class Fields, page 240.

The following are additional Call class fields.

line

Description

The line object that is associated with this call.

Type: line object.

See Also

Chapter 12, "JSMCAPI Classes," Line Class Constructor, page 201

statistics

Description

Statistics associated with the call.

Type: CallStatistics object.

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.

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 call data.

Type: string.

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.

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.

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 12, "JSMCAPI Classes," Task Class Constructor, page 240.

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.

Returns

A Chat object.

Chat Class Fields

The Chat class inherits the following fields from Task class:

- caseid
- cost
- customerid
- group
- id
- onStat

- priority
- type

  The task type for chat that is Task.TYPE_CHAT.
- urlAbs
- urlRel

See Chapter 12, "JSMCAPI Classes," Task Class Fields, page 240.

The following are the additional Chat class fields.

address

Description

The address containing the agent-to-agent chat.

Type: A2AChatAddress object.

See Also

Chapter 12, "JSMCAPI Classes," A2AChatAddress Class Constructor, page 157

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 12, "JSMCAPI Classes," AppData Class Constructor, page 161

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.

Chat Class Method

The Chat class extends the Task class.

See Chapter 12, "JSMCAPI Classes," Task Class Constructor, page 240.

The following is the Chat class method.

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 12, “JSMCAPI Classes,” _UQAddress Class Constructor, page 148.

The following is the ChatAddress class constructor.

ChatAddress

Syntax

```
ChatAddress()
```

Description

Handles the creation of customer chat tasks.

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 12, “JSMCAPI Classes,” _Address Class Fields, page 147.

The ChatAddress class inherits the following field from the _UQAddress class:

tasks

See Chapter 12, “JSMCAPI Classes,” _UQAddress Class Fields, page 148.

ChatAddress Class Methods

The ChatAddress class inherits the following methods from the _UQAddress class:

- acceptTask
- dequeueTask

See Chapter 12, “JSMCAPI Classes,” _UQAddress Methods, page 148.

ChatAddress Callback Event Methods

The ChatAddress class inherits the onError callback event method from the _Address class.

See Chapter 12, “JSMCAPI Classes,” _Address Class Callback Event Method, page 147.

The ChatAddress class inherits the following callback event methods from the _UQAddress class:

- onAccepted
- onAcceptingTask
- onDequeueingTask
- onNotify
- onTaskAdded
- onTaskRemoved
- onUnassigned

See Chapter 12, “JSMCAPI Classes,” _UQAddress Class Fields, page 148.

Email Class Constructor

The Email class extends the Task class.

See Chapter 12, “JSMCAPI Classes,” Task Class Constructor, page 240.

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.

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 12, "JSMCAPI Classes," Task Class Fields, page 240.

The following are the additional Email class fields:

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 12, "JSMCAPI Classes," Task Class Fields, page 240

Email Class Method

The following is the Email 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.

EmailAddress Class Constructor

The EmailAddress class extends the _UQAddress class.

See Chapter 12, "JSMCAPI Classes," _UQAddress Class Constructor, page 148.

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 12, "JSMCAPI Classes," _Address Class Fields, page 147.

The EmailAddress class inherits the tasks field from the _UQAddress class.

See Chapter 12, "JSMCAPI Classes," _UQAddress Class Fields, page 148.

EmailAddress Class Methods

The EmailAddress class inherits the following fields from the _UQAddress class:

- acceptTask
- dequeueTask

See Chapter 12, "JSMCAPI Classes," _UQAddress Methods, page 148.

EmailAddress Callback Event Methods

The EmailAddress class inherits the onError callback event method from the _Address class.

See Chapter 12, "JSMCAPI Classes," _Address Class Callback Event Method, page 147.

The EmailAddress class inherits the following callback event methods from the _UQAddress class:

- onAccepted
- onAcceptingTask
- onDequeueingTask
- onNotify
- onTaskAdded
- onTaskRemoved
- onUnassigned

See Chapter 12, "JSMCAPI Classes," _UQAddress Class Callback Event Methods, page 149.

Extension Class Constructor

The Extension class extends the _Address class.

See Chapter 12, "JSMCAPI Classes," _Address Class Constructor, page 146.

The following is the Extension class constructor.

Extension

Syntax

```
Extension(numOfLines)
```

Description

The Extension object describes the CTI address.

Parameters

Parameter	Description
numOfLines	Specify 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 12, “JSMCAPI Classes,” _Address Class Fields, page 147.

The following are the Extension class fields.

forwardMode

Description

The forward mode of the extension.

Type: forwardMode object.

See Also

Chapter 12, “JSMCAPI Classes,” ForwardMode Class Constructor, page 185

isDnd

Description

Flag for Do Not Disturb.

Type: boolean.

lines

Description

An associative array of line objects.

Type: associative array.

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.

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

None.

cancelForwardSet

Syntax

cancelForwardSet(*reason*)

Description

Cancel the forward.

Parameters

Parameter	Description
reason	The reason for cancelling the forward.

Returns

None.

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

None.

getDialingLine

Syntax

getDialingLine()

Description

Get the dialing line object.

Parameters

None

Returns

Returns a line object. Null if there is no dialing line.

See Also

Chapter 12, "JSMCAPI Classes," Line Class Constructor, page 201

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. 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.

Returns

Returns a Line object. Null if there is no line with that call.

See Also

Chapter 12, "JSMCAPI Classes," Line Class Constructor, page 201

getOffHookLine

Syntax

getOffHookLine()

Description

Get the offhook line object.

Parameters

None.

Returns

Returns a Line object. Null if there is no offhook line.

See Also

Chapter 12, "JSMCAPI Classes," Line Class Constructor, page 201

setDnd

Syntax

setDnd(*event*)

Description

Fires when setting DND (do not disturb).

Parameters

Parameter	Description
event	The event object.

Returns

None.

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*)

Description

Fires when DND is selected.

onForwardCanceled

Syntax

onForwardCanceled(*event*)

Description

Fires on cancellation of forwarding.

onForwarded

Syntax

onForwarded(*event*)

Description

Fires when the extension is forwarded.

onForwarding

Syntax

onForwarding(*event*)

Description

Fires when forwarding.

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.

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.

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.

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.

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.

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 12, "JSMCAPI Classes," _Address Class Fields, page 147.

The GenericAddress class inherits the tasks field from the _UQAddress class.

See Chapter 12, "JSMCAPI Classes," _UQAddress Class Fields, page 148.

GenericAddress Class Methods

The GenericAddress class inherits the following methods from the _UQAddress class:

- acceptTask
- dequeueTask

See Chapter 12, "JSMCAPI Classes," _UQAddress Methods, page 148.

GenericAddress Class Callback Event Methods

The GenericAddress class inherits the onError callback event method from the _Address class.

See Chapter 12, "JSMCAPI Classes," _Address Class Callback Event Method, page 147.

The GenericAddress class inherits the following callback event methods from the _UQAddress class:

- onAcceptingTask
- onDequeueingTask
- onTaskAdded
- onTaskRemoved
- onNotify
- onAccepted
- onUnassigned

See Chapter 12, "JSMCAPI Classes," _UQAddress Class Callback Event Methods, page 149.

GenericTask Class Constructor

The GenericTask class extends the Task class.

The following is the GenericTask class constructor

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

See Chapter 12, “JSMCAPI Classes,” Task Class Fields, page 240.

The following are the additional GenericTask class fields.

agentId

Description

The agent's ID.

Returns a string.

appData

Description

The application data that is provided to the client with the notification event.

Type: AppData object.

See Also

Chapter 12, "JSMCAPI Classes," AppData Class Constructor, page 161

customerName

Description

The customer username.

Type: string.

groupId

Description

The group ID.

Type: string.

question

Description

The question for this generic task.

Type: string.

subject

Description

The subject of the generic task.

Type: string.

GenericTask Class Method

The following is the GenericTask class method.

getUrl

Syntax

getUrl(*defaultUrl*)

Description

Returns the URL for the given task.

Parameters

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";

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.

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

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.

GLOBALS class methods

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.

isValid

Syntax

```
isValid(obj)
```

Description

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.

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.

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 12, "JSMCAPI Classes," GroupStatistics1 Class Constructor, page 197

statistics1

Description

The group statistics for UQ server.

Type: GroupStatistics1 object.

See Also

Chapter 12, "JSMCAPI Classes," GroupStatistics1 Class Constructor, page 197

statistics2

Description

The group statistics for UQ server.

Type: GroupStatistics2 object.

See Also
Chapter 12, "JSMCAPI Classes," GroupStatistics2 Class Constructor, page 200

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.

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 there is task added to this group.

onTaskRemoved

Syntax
onTaskRemoved(*event*)

Description

Fires when a task is removed from this group.

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.

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.

queuedWaitTime

Description

The average wait time, in seconds, of a queued task.

relativeQueueLoad

Description

Relative queue load.

GroupStatistics1 Class Constructor

The following is the GroupStatistics1 class constructor.

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.

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.

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.

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.

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.

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.

Line Class Fields

The following are the Line class fields.

call

Description

The Call object on the line.

Type: Call object.

See Also

Chapter 12, "JSMCAPI Classes," Call Class Constructor, page 165

caps

Description

The capabilities of the line.

Type: LineCaps object.

See Also

Chapter 12, "JSMCAPI Classes," LineCaps Class Constructor, page 217

connectionid

Description

The call connection ID on the line.

Type: string.

id

Description

The line ID.

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.

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

None.

answer

Syntax

answer(*reason*)

Description

Answer a call.

Parameters

Parameter	Description
reason	The reason to answer a call.

Returns

None.

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

None.

clear

Syntax

clear(*reason*)

Description

Clear a conference call.

Parameters

Parameter	Description
reason	The reason to clear the conference.

Returns

None.

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

None.

conference

Syntax

conference(*destination*, *reason*, *userdata*, *calldata*)

Description

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

None.

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

None.

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

None.

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.

Returns

None.

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.

getDnis

Syntax

```
getDnis()
```

Description

Get the DNIS.

Parameters

None.

Returns

Returns a string.

getReferenceId

Syntax

```
getReferenceId()
```

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 popup.

Parameters

Parameter	Description
defaultUrl	The default popup URL.

Returns

Returns a string.

grabCall

Syntax

```
grabCall(destination, reason)
```

Description

Grab a call from the queue.

Parameters

Parameter	Description
destination	The destination to be grabbed.
reason	The reason for the grab.

Returns

None.

hold

Syntax

hold(*reason*)

Description

Hold a call.

Parameters

Parameter	Description
reason	The reason to hold the call.

Returns

None.

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

None.

reconnect

Syntax

reconnect(*reason*)

Description

Reconnect to the original party during two-step transfer/conference.

Parameters

Parameter	Description
reason	The reason to reconnect.

Returns

None.

release

Syntax

release(*reason*)

Description

Release a call.

Parameters

Parameter	Description
reason	The reason to release the call.

Returns

None.

retrieve

Syntax

retrieve(*reason*)

Description

Retrieve a held call.

Parameters

Parameter	Description
reason	The reason to retrieve the call.

Returns

None.

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

None.

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.
reason	The reason to transfer.
userdata	The user data to be attached.
calldata	The call data to be modified.

Returns

None.

updateCallData

Syntax

updateCallData(*calldata, reason*)

Description

Update call data to a call.

Parameters

Parameter	Description
calldata	The call data to update.
reason	The reason to update.

Returns

None.

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.

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.

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.

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.

onOffHook

Syntax

onOffHook(*event*)

Description

Fires when the line is off hook.

onOnHook

Syntax

onOnHook(*event*)

Description

Fires when the line is on hook.

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.

onReconnecting

Syntax

onReconnecting(*event*)

Description

Fires when reconnecting the call.

onReleasing

Syntax

onReleasing(*event*)

Description

Fires when releasing the call.

onRetrieving

Syntax

onRetrieving(*event*)

Description

Fires when retrieving the call.

onRinging

Syntax

onRinging(*event*)

Description

Fires when there is an incoming call.

onTalking

Syntax

onTalking(*event*)

Description

Fires when the call is established.

onTransfering

Syntax

onTransfering(*event*)

Description

Fires when transferring the call.

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.

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.

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.

canPark

Description

The line has park capability.

Type: boolean.

canReconnect

Description

The line has reconnect capability.

Type: boolean.

canRelease

Description

The line has release capability.

Type: boolean.

canRetrieve

Description

The line has retrieve capability.

Type: boolean.

canTransfer

Description

The line has transfer capability.

Type: boolean.

canTransferMute

Description

The line has transfer mute capability.

Type: boolean.

canUpdateCallData

Description

The line has update call data capability.

Type: boolean.

MCEvent Class Constructor

The following is the MCEvent class constructor.

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 12, “JSMCAPI Classes,” Extension Class Constructor, page 177

group

Description

The group associated with the event.

Type: Group object.

See Also

Chapter 12, “JSMCAPI Classes,” Group Class Constructor, page 193

reason

Description

The reason associated with the event.

Type: Reason object.

See Also

Chapter 12, “JSMCAPI Classes,” Reason Class Constructor, page 226

user

Description

The user associated with the event.

Type: User object.

See Also

Chapter 12, “JSMCAPI Classes,” User Class Constructor, page 242

MediaType Class Constructor

The following is the MediaType class constructor.

MediaType

Syntax

```
MediaType()
```

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_EMAIL	The email media type.
MT_VOICE	The voice media type (CTI).

PSMC Class Constructor

The following is the PSMC class constructor.

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 12, "JSMCAPI Classes," RenServer Class Constructor, page 227

servers

Description

An associative array the different Servers objects, indexed by server ID.

Type: associative array.

See Also

Chapter 12, "JSMCAPI Classes," Server Class Constructor, page 228

sessions

Description

An associative array that holds the different Session objects, indexed by server ID.

Type: associative array.

See Also

Chapter 12, "JSMCAPI Classes," Session Class Constructor, page 231

PSMC Class Methods

The following are the PSMC class methods.

closeSession

Syntax

closeSession(*id*)

Description

Closes the specified Session and Server object and deletes them.

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.

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 12, "JSMCAPI Classes," Call Class Constructor, page 165

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 12, "JSMCAPI Classes," Line Class Constructor, page 201

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.

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*)

Description

The Reason object carries the reason code and reason description.

Parameters

Parameter	Description
code	The reason code.
desc	The reason description.

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.

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.

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.

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.

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.
TYPE_UQ	The server is a UQ 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 12, "JSMCAPI Classes," GLOBALS Class Fields, page 190

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.

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.

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.

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.

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 12, "JSMCAPI Classes," User Class Constructor, page 242

Session Class Methods

The following are the Session class methods.

close

Syntax
```
close()
```

Description

Close the session with the server.

Parameters

None.

Returns

None.

open

Syntax
```
open()
```

Description

Open the session with the server.

Parameters

None.

Returns

None.

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.

Returns

None.

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

None.

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

None.

registerUser

Syntax

registerUser(*user*)

Description

Register the user to the server so that the server will report all those events happened on this user.

Parameters

Parameter	Description
user	The user to be registered.

Returns

None.

statPublish

Syntax

statPublish(*userStat, groupStat*)

Description

Sets the Statistics publishing levels for the UQ server.

Parameters

Parameter	Description
userStat	Either 0, 1, or 2 for UserStatistics publishing. Specify 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. Specify 0 for no statistics publishing, 1 for Statistics1 publishing, and 2 for Statistics2 and Statistic1 publishing.

Returns

None.

unregisterAddress

Syntax

unRegisterAddress(*address*)

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

None.

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

unregisterGroup

Syntax

unregisterGroup(*group*)

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

None.

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

None.

Session Class Callback Event Methods

The following are the Session class callback event methods.

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.

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.

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

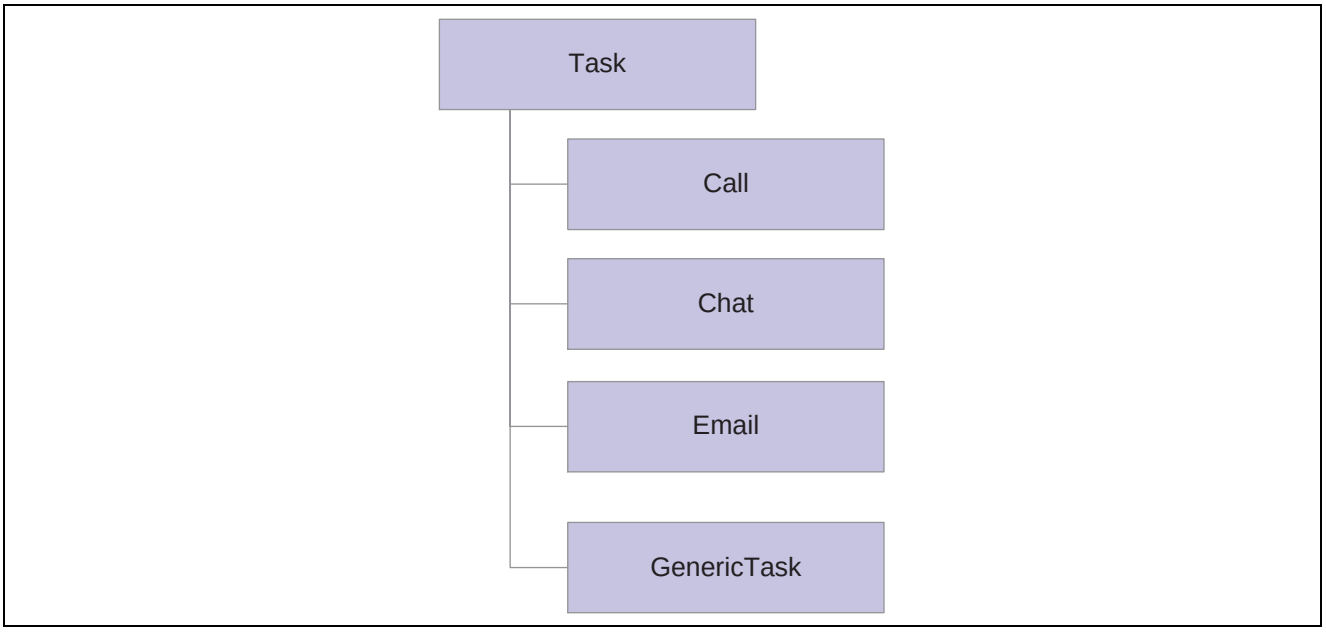
onUserUnregistered(*event*)

Description

Fires when the user gets unregistered by the session.

Task Class Hierarchy

The Task class can be extended by other subclasses. The following flowchart shows the different subclasses, and how they interrelate.



Task class hierarchy

See Also

Chapter 12, “JSMCAPI Classes,” Call Class Constructor, page 165

Chapter 12, “JSMCAPI Classes,” Chat Class Constructor, page 170

Chapter 12, “JSMCAPI Classes,” Email Class Constructor, page 174

Chapter 12, “JSMCAPI Classes,” GenericAddress Class Constructor, page 186

Task Class Constructor

The following is the Task class constructor.

Task

Syntax

```
Task()
```

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.

group

Description

The associated group.

Type: string.

id

Description

The associated ID.

Type: string.

onStat

Description

The associated onStat.

Type: string.

priority

Description

The associated priority.

Type: string.

type

Description

The associated type.

Type: string, one of the following:

Value	Description
TYPE_A2ACHAT	The A2AChat type.
TYPE_CHAT	The chat type
TYPE_CTI	The CTI type.

Value	Description
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 12, "JSMCAPI Classes," GLOBALS Class Fields, page 190

urlAbs

Description

The associated absolute URL.

Type: urlAbs object.

urlRel

Description

The associated relative URL.

Type: urlRel object.

User Class Constructor

The User class extends the _User class.

See Chapter 12, "JSMCAPI Classes," _User Class Constructor, page 151.

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.

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.

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.

User Class Methods

The following are the User class methods.

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.
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

None.

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

None.

isMediaEnabled

Syntax

```
isMediaEnabled(mediaType, group)
```

Description

Determines if the specified media type is enabled.

Parameters

Parameter	Description
mediaType	Specify the media type.
group	Specify 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.

Returns

None.

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

None.

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.
extension	The extension.
reason	The reason for the lout.

Returns

None.

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

None.

register

Syntax

register(*extension, reason*)

Description

Register an extension for the user.

Parameters

Parameter	Description
extension	The extension to be registered.
reason	The reason to register the extension.

Returns

None.

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 that the user is changing state with.
reason	The reason to set not ready.

Returns

None.

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 that the user is changing presence with.
reason	The reason to set the presence.

Returns

None.

setPresenceUq

Syntax

setPresenceUq(*group*, *presenceText*, *reason*)

Description

Set the Presence for UQ servers.

Parameters

Parameter	Description
group	The group (queue) the user is changing their presence on.
presenceText	The new presence.
reason	The reason to set the new presence.

Returns

None.

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 that the user is changing state with.
reason	The reason to set the state.

Returns

None.

setWorkNotReady

Syntax

setWorkNotReady(*group*, *presence*, *extension*, *reason*)

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 that the user is changing state with.
reason	The reason to set work not ready.

Returns

None.

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 that the user is changing state with.
reason	The reason to set work ready.

Returns

None.

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

None.

User Class Callback Event Methods

The following are the User class callback event methods.

onCapabilitiesChanged

Syntax

onCapabilitiesChanged(*event*)

Description

Fires when capabilities changed.

onCtiBusy

Syntax

onCtiBusy(*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 user is dropped.

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.

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.

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.

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*)

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*)

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*)

Description

Fires when unregistering the address.

onWorkNotReady

Syntax

onWorkNotReady(*event*)

Description

Fires when the user's state to changes 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.

Parameters

Parameter	Description
strCaps	The user capabilities.

Returns

A UserCaps object.

UserCaps Class Fields

The following are the UserCaps class fields

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.

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.

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.
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.

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.

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.

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.

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.

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.

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.

CHAPTER 13

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.
- 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 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 *Enterprise PeopleTools 8.46 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 email attachments, including:
  - An attachments repository running on 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) 1.3.1 or later.

  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.

Handling Email

This section describes some factors to consider when handling email using PeopleSoft MultiChannel Framework.

Time Zone Offsets

Set the values of email sent time timezone offset (in minutes) and receive time timezone offset (in minutes) whenever possible. The default value is 800 which indicates timezone information not available. When available the values range from +720 to -720.

Time Zone Offsets

Set the values of email sent time timezone offset (in minutes) and receive time timezone offset (in minutes) whenever possible. The default value is 800 which indicates timezone information not available. When available the values range from +720 to -720.

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.

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	Object 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

Enterprise PeopleTools 8.46 PeopleBook: Integration Broker, "Managing Integration Gateways"

Configuring the Gateway

Access the Gateways page.

To configure the gateway:

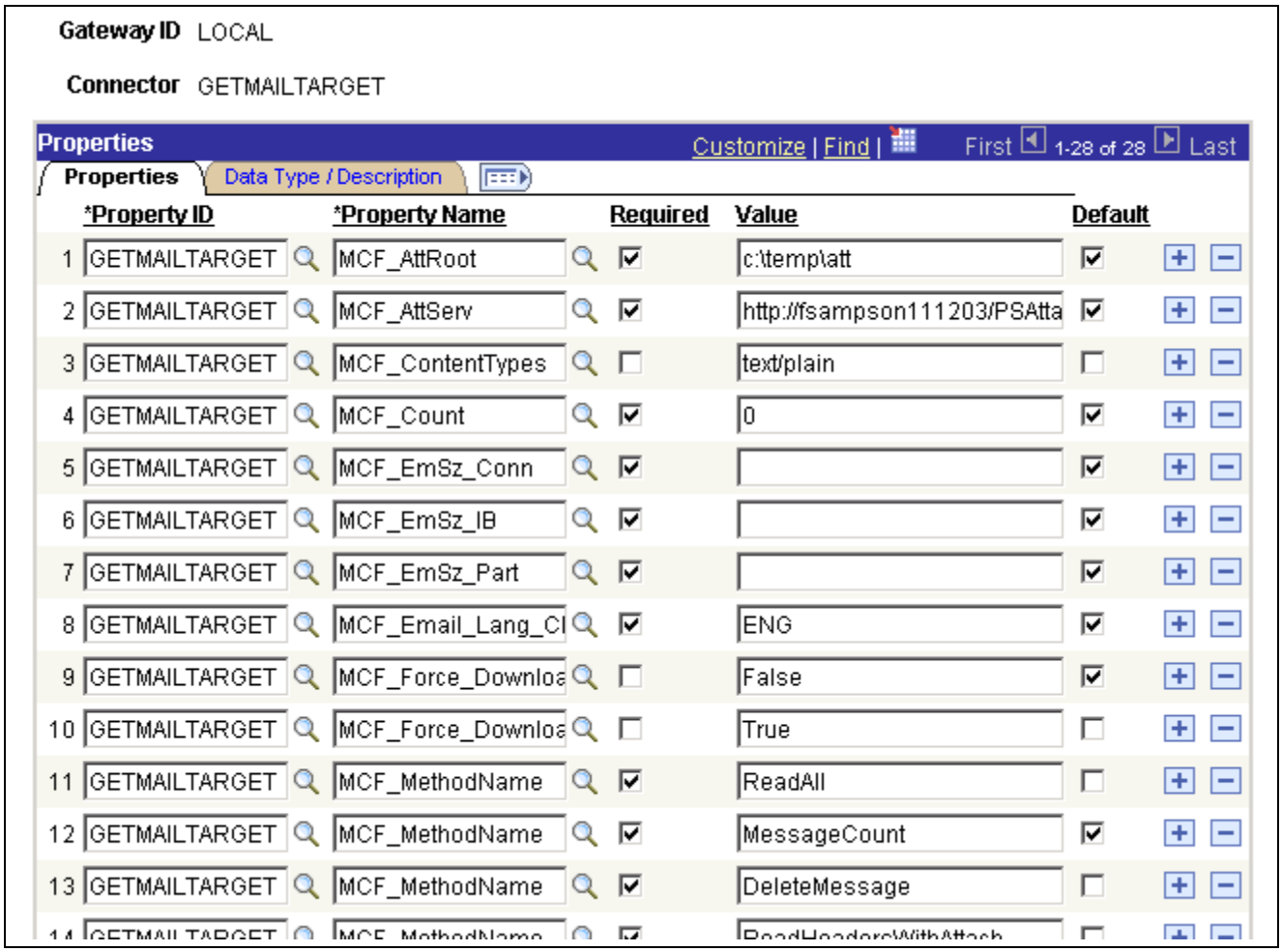
1. Create a local gateway.
2. Specify the gateway URL in the form http://<gatewayservername>/PSIGW/PeopleSoftListeningConnector, where <gatewayservername> is the fully qualified host name for the gateway machine.

   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.



Connector Properties 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:

Property Name	Default Value	Description
MCF_AttRoot	*c:\temp\att*	Defines the location of the attachment repository on the gateway. 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>. **Note.** The user name or mailbox portion of the attachment root path may include encoded characters. 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. Attachment security is managed at a user and role level using the PT_MCF_EMAIL application package. Users can configure attachment folder paths using metastrings to enable multiple folders based on several variables: • *%OPRID%:* PeopleSoft user ID. • *%DBNAME%:* database name. • *%CURRDATE%:* current date. • *%CURRHOUR%:* current hour.
MCF_AttServ	*http://<machine_ name>.<domain_name> /PSAttachServlet/ps/*	Defines the location of the MCF attachment repository servlet, located on the gateway web server. Attachment relative URLs are appended to this address to create a fully qualified URL to reference an attachment in the repository. **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.

Property Name	Default Value	Description
MCF_ContentTypes	*text/plain*	Specifies email content types that will be stored in the database. Content types not specified will be stored as attachments. Specify content types separated by commas. For example, you can specify *text/html* and *text/xml* to enable storage in the database of HTML email and email containing XML. The content type *text/plain* is always stored in the database, regardless of values specified for MCF_ContentTypes. Do not include binary encoded content types in the content types list unless the application can handle base64 encoded text.
MCF_Count	*0*	Defines the default number of emails retrieved from the mail server unless otherwise specified in the SyncRequest. 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. 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. 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. See *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference*, “Mail Classes”.

Property Name	Default Value	Description
MCF_EmSz_IB	*500000*	Defines the PeopleSoft Integration Broker threshold, in bytes. 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. 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. See *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference*, "Mail Classes".
MCF_EmSz_Part	*30000*	Defines the text part threshold, in bytes. 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. 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. 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.

Property Name	Default Value	Description
MCF_MethodName	*MessageCount*	Defines the methods associated with the application class package. 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. See *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference*, "Mail Classes".
MCF_Password	None	Defines the default password for the mailbox unless otherwise specified in the SyncRequest. See *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference*, "Mail Classes".
MCF_Port	*110*	Defines the mail server port to be used. 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	*POP3*	Defines the protocol used by the connector to access emails on the mail server. Supported values are *POP3* or *IMAP4*.
MCF_Quarantine	*Quarantine*	Defines a quarantine folder for email meeting the following criteria: • Connector size overflow. • Unsupported encoding. • Unknown Java exception. • JavaMail content error. • No attachment repository. • Mail parse exception. The quarantine folder is created for each user account and must be managed by each email user, not by system administrators. **Note.** Folders are supported only on IMAP4 mail servers.

Property Name	Default Value	Description
MCF_Server	None	Defines the fully qualified host name of the mail server. This is the default fully qualified host name of the mail server, unless otherwise specified in the SyncRequest. For example, bigserver.peoplesoft.com. See *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference*, "Mail Classes".
MCF_User	None	Defines the user name for the email account (mailbox) being accessed. This is the default user name used unless otherwise specified in the SyncRequest. For example, if emails are addressed to support@peoplesoft.com, the user name is *support*.
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.

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.

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.

PeopleSoft provides email sample pages to demonstrate 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.

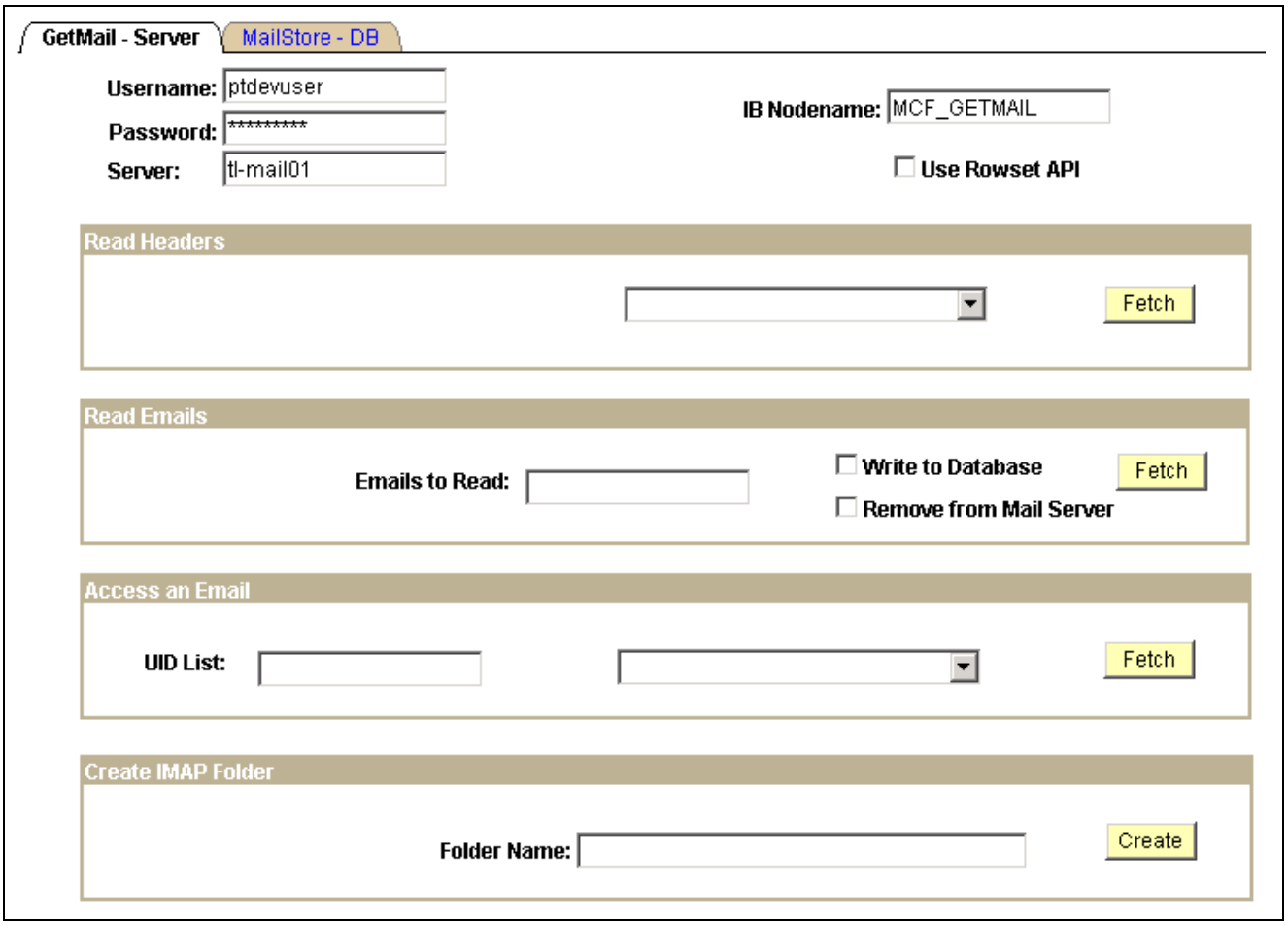
Note. Using the email sample pages requires access to a mail server.

Pages Used to Demonstrate the Email Channel

Page Name	Object 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.



GetMail - Server page

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 specify on the GetMail - Server page.

Username and **Password**	Specify a valid username and password for an account on the mail server.
Server	Specify the mail server.
IB Nodename (integration broker node name)	Specify 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

Specify the number of emails to read, select other parameters, then click Fetch to read emails.

Emails to Read Specify 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

Specify an email UID list, select a method from the drop-down list box, then click Fetch to access the selected emails.

UID List (unique identifier list) Specify 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.

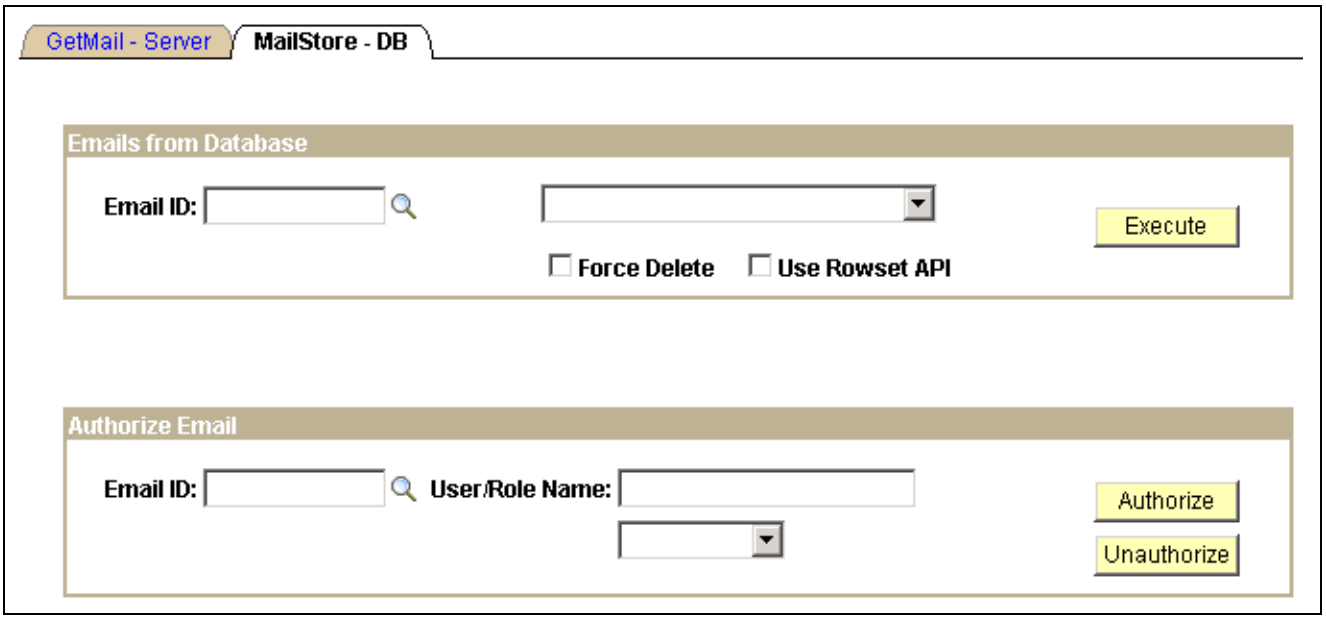
Create IMAP Folder

Create an email folder; only valid for IMAP4 mail servers.

Folder Name Specify the name of the folder to create.

Using the MailStore - DB Page

Access the MailStore - DB page.



MailStore - DB page

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 MultiChannel Framework.

Email ID Specify the PeopleSoft email ID (not to be confused with the UID) with which the email was saved to the database.

Emails from Database

Specify an email ID, select an action from the drop-down list box, and then click Execute to perform the selected action.

Select from:

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 specifying *Delete Email from Database* to force the deletion even if there is an error deleting any 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

Specify an email ID, a user role or name, and click Authorize or Unauthorize to perform the specified action.

Email ID Specify the PeopleSoft email ID (not to be confused with the UID) with which the email was saved to the database.

User/Role Name Specify a PeopleSoft user ID or role to be authorized or unauthorized to view the attachment associated with the selected email.

After specifying an ID or role, select *User* or *Role,* as appropriate, from the drop-down list box.

See Also

Chapter 11, “Developing for PeopleSoft MultiChannel Framework,” Using the Email Sample Page, page 127

CHAPTER 14

Configuring Instant Messaging

This chapter provides an overview of instant messaging and discusses how to:

- Configure instant messaging in PeopleSoft MultiChannel Framework.
- Use the instant messaging sample pages.

See Also

Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference, "MCFIMInfo Class"

Understanding Instant Messaging

PeopleSoft has extended 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 AOL Instant Messenger, IBM/Lotus Sametime, or Yahoo! Messenger. This enables an external user to participate in a business transaction through instant messaging.

PeopleSoft MultiChannel Framework enables instant messaging through public networks (AOL Instant Messenger and Yahoo! Messenger) and through an enterprise Lotus Sametime Connect network.

You can enable a user to initiate an instant messaging session from any page using either a button or PeopleCode.

PeopleSoft does not supply the client for any supported instant messaging software. Each user must install the appropriate client software. However, presence detection for AOL Instant Messenger and Yahoo! Messenger users functions even without installed client software.

Note. The instant messaging presence detection integration in PeopleTools (the "IM Integration") 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, PeopleSoft is not responsible for the availability or performance of any of the IM Networks or for any change in an IM vendor's protocol which may render the IM Integration inoperable.

Configuring Instant Messaging Servers

The supported public instant messaging networks, Yahoo! Messenger and AOL Instant Messenger, 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.

Configuring Instant Messaging

To configure instant messaging, use the Instant Messaging Configuration (MCF_IM_CFG_CMP) component.

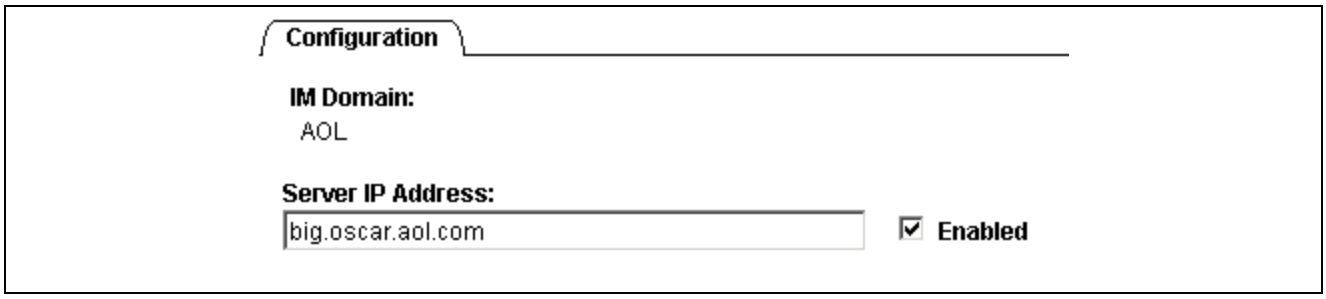
This section discusses how to configure instant messaging.

Page Used to Configure Instant Messaging

Page Name	Object Name	Navigation	Usage
Configuration	MCF_IM_CFG_PG	PeopleTools, MultiChannel Framework, Instant Messaging, Configuration	Specify or edit server IP address for instant messaging domains.

Configuring Instant Messaging

Configure instant messaging by enabling domains.



Configuration page

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 AOL.

The server addresses for Yahoo and AOL are preconfigured. However, specify the address for your Sametime Connect server.

Enabled Select Enabled to enable the configured instant messaging service.

If the enabled flag is unselected the network is still available, but the buttons will always show presence as offline.

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:

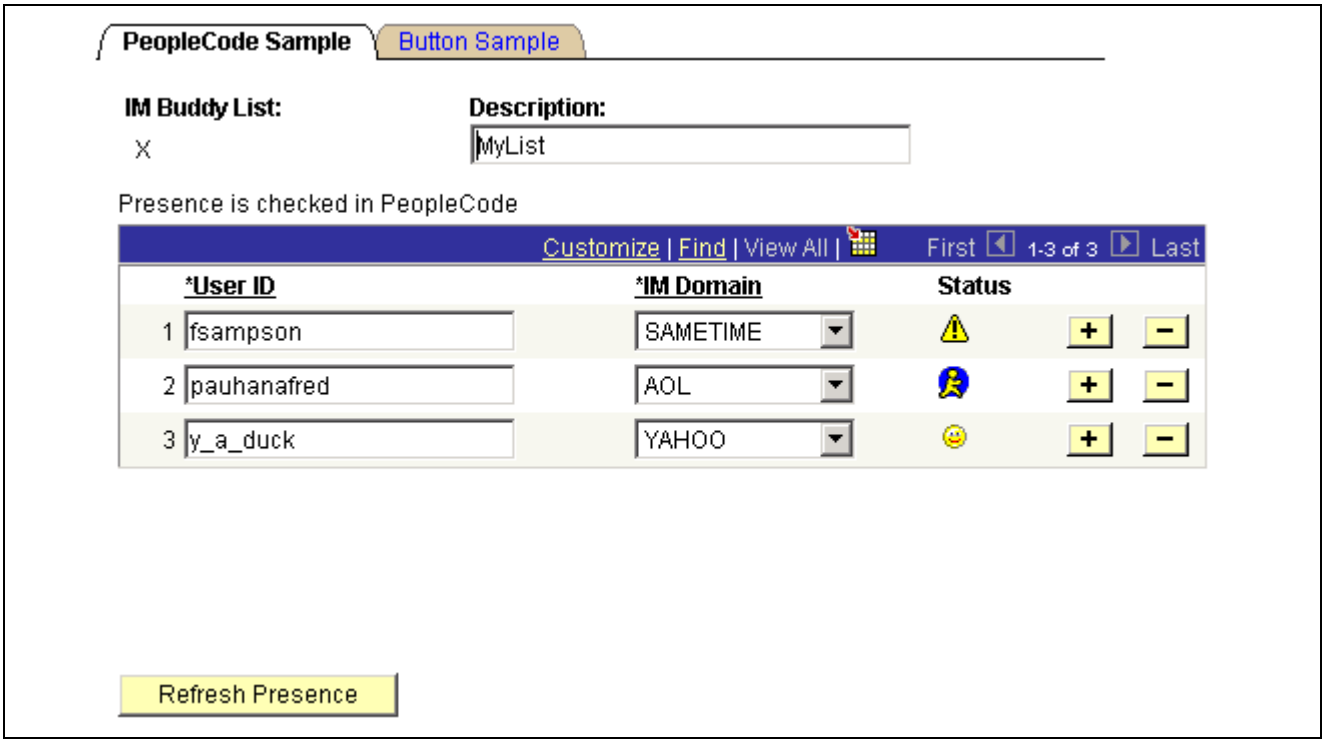
- Use the PeopleCode Sample page.
- Use the Button Sample page.

Pages Used to Test Instant Messaging

Page Name	Object 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.
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

Access the PeopleCode Sample page.



PeopleCode Sample page

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 *Enterprise PeopleTools 8.46 PeopleBook: PeopleCode API Reference*, "MCFIMInfo Class".

IM Buddy List An identifier for the list.

Description A description for the list.

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.

Status An icon displays the presence status of the specified user. The icons vary with the instant messaging service.

Click the icon to initiate an instant messaging session with the selected user. The user must be online.

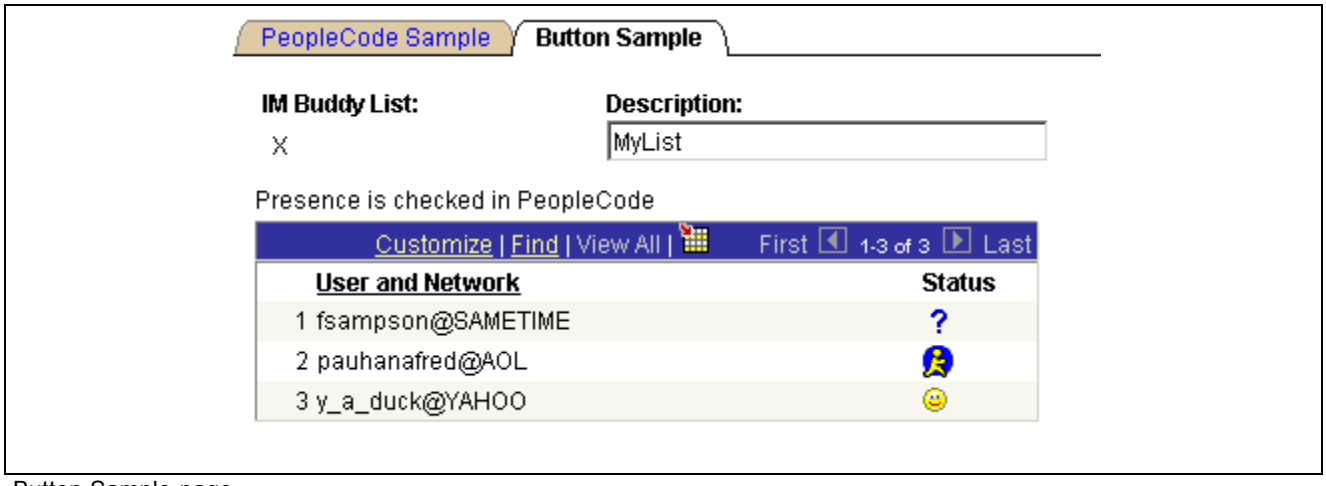
Note. Yahoo! and AOL do not update presence automatically. Use the Refresh Presence button to check the current status of Yahoo! and AOL users.

Refresh Presence Click the Refresh Presence button to check the status of AOL and Yahoo users.

Note. PeopleCode does not support presence detection in Sametime, so the Sametime user status does not display on the PeopleCode Sample page. Use the instant messaging button if you intend to check presence of or connect to Sametime users.

Using the Button Sample Page

Access the Button Sample page.



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 *Enterprise PeopleTools 8.46 PeopleBook: PeopleSoft Application Designer*, "Using Page Controls," Specifying Destination Types.

IM Buddy List An identifier for this list.

Description A description for this list.

User and Network User ID and instant messaging service ID.

Status 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. If the AOL icon does not display correctly, be sure that "Use HTTP 1.1" is selected in Microsoft Internet Explorer advanced internet options. If you must select "Use HTTP 1.1," be sure to close the current browser session and open another before continuing.

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, and callback event methods, and returns associated with each class.

_Address

This section lists the constructor, fields, methods, and 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 Methods	Returns
onError(*event*)	None.

_UQAddress

This section lists the constructor, fields, methods, and 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 285.

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 285.

Callback Event Methods	Returns
onAccepted(*event*)	None.
onAcceptingTask(*event*)	None.
onDequeueingTask(*event*)	None.
onNotify(*event*)	None.
onTaskAdded(*event*)	None.
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
statistics1	Object
statistics2	Object

A2AChat

This section lists the constructor, fields, methods, and 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 285.

Constructor

Constructor	Returns
A2AChat(*event*, *address*, *chatType*)	A2AChat object.

Fields

Field	Type
address	Object
agentId	String
agentName	String
appData	Object
chatType	String
id	String
jr	String of the PS_JR constant.
type	String of the following constants: • TYPE_ANSWER • TYPE_CONSULT
uniqueueId	String

Method

Method	Returns
getUrl(*defaultUrl*)	String

A2AChatAddress

This section lists the constructor, fields, methods, and 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 285.

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.

See Appendix A, "JSMCAPI Quick Reference," _Address, page 285.

Callback Event Methods	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	String
averageHoldDuration	String
callsHandled	String
data	Associative array

Field	Type
unavailableDuration	String
waitDuration	String

AppData

This section lists the constructor, fields, methods, and 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

Method

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 286.

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 286.

Callback Event Methods

Callback Event Methods	Returns
onStat1(*event*)	None
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 314.

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 314.

Field	Type
line	Object
statistics	Object
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

Method

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

Fields

Field	Type
data	Associative array
holdTime	String
queueTime	String
talkTime	String

Chat

This section lists the constructor, fields, method, and returns (if applicable) associated with this class.

The Chat class extends the Task class.

See Appendix A, "JSMCAPI Quick Reference," Task, page 314.

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 314.

Field	Type
address	Object
agentId	String
appData	Object
customerName	String
groupId	String
question	String
subject	String

Method

Method	Returns
getUrl(*defaultUrl*)	String

ChatAddress

This section lists the constructor, fields, methods, and 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 285.

Constructor

Constructor	Returns
ChatAddress()	ChatAddress object

Fields

The ChatAddress class inherits the following fields from the _Address class:

- id
- caps

See Appendix A, “JSMCAPI Quick Reference,” _Address, page 285.

The ChatAddress class inherits the tasks field from the _UQAddress class.

See Appendix A, “JSMCAPI Quick Reference,” _UQAddress, page 285.

Methods

The ChatAddress class inherits the following methods from the _UQAddress class:

- acceptTask
- dequeueTask

See Appendix A, “JSMCAPI Quick Reference,” _UQAddress, page 285.

Callback Event Methods

The ChatAddress class inherits the onError callback event method from the _Address class.

See Appendix A, “JSMCAPI Quick Reference,” _UQAddress, page 285.

The ChatAddress 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 285.

Email

This section lists the constructor, fields, method, and returns (if applicable) associated with this class.

The Email class extends the Task class.

See Appendix A, “JSMCAPI Quick Reference,” Task, page 314.

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 314.

Field	Type
agentId	Object
appData	Object
customerName	String

Method

Method	Returns
getUrl(*defaultUrl*)	String

EmailAddress

This section lists the constructor, fields, methods, and 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 285.

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 285.

The EmailAddress class inherits the tasks field from the _UQAddress class.

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 285.

Methods

The EmailAddress class inherits the following methods from the _UQAddress class:

- acceptTask
- dequeueTask

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 285.

Callback Event Methods

The EmailAddress class inherits the onError callback event method from the _Address class.

See Appendix A, "JSMCAPI Quick Reference," _Address, page 285.

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 285.

Extension

This section lists the constructor, fields, methods, and 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 285.

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 285.

Field	Type
forwardMode	Object
isDnd	Boolean
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 285.

Callback Event Methods	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

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

Field	Type
mode	String of the following constants: • BUSY • BUSYNOANSWER • DEFAULT • NOANSWER • NONE • UNCONDITIONAL

GenericAddress

This section lists the constructor, fields, methods, and 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 285.

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 285.

The GenericAddress class inherits the tasks field from the _UQAddress class.

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 285.

Methods

The GenericAddress class inherits the following methods from the _UQAddress class:

- acceptTask
- dequeueTask

See Appendix A, "JSMCAPI Quick Reference," _UQAddress, page 285.

Callback Event Methods

The GenericAddress class inherits the onError callback event method from the _Address class.

See Appendix A, “JSMCAPI Quick Reference,” _Address, page 285.

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 285.

GenericTask

This section lists the constructor, fields, methods, and 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 314.

Constructor

Constructor	Returns
GenericTask(*event*)	GenericTask object

Field

The GenericTask 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 314.

Field	Type
agentId	String
appData	Object
customerName	String
groupId	String
question	String
subject	String

Method

Method	Returns
getUrl(*defaultUrl*)	String

GLOBALS

This section lists the fields and methods associated with this class.

Fields

Field	Field Type
A2AChat.PS_JR	Constant
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

Group

This section lists the constructor, fields, and callback event methods, and returns (if applicable) associated with this class.

Constructor

Constructor	Returns
Group()	Group object

Fields

Field	Type
id	String
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
numberOfAbandoned	String
numberOfLoggedIn	String
numberOfQueued	String
queuedWaitTime	String
relativeQueueLoad	String

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
numEscalation	String
numOverflow	String
numTaskAccepted	String

Field	Type
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

Line

This section lists the constructor, fields, methods, and 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
state	String of the following constants: • ST_DIALING • ST_DROPPED • ST_HELD • ST_IDLE • ST_OFFHOOK • ST_RINGING • ST_TALKING

Methods

Method	Returns
alternate(*reason*)	None
answer(*reason*)	None
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

Method	Returns
getUrl(*defaultUrl*)	String
grabCall(*destination*, *reason*)	String
hold(*reason*)	String
park(*destination*, *reason*)	None
reconnect(*reason*)	None
release(*reason*)	None
retrieve(*reason*)	None
transfer(*destination*, *reason*, *userdata*, *calldata*)	None
transferMute(*destination*, *reason*, *userdata*, *calldata*)	None
updateCallData(*calldata*, *reason*)	None

Callback Event Methods

Callback Event Methods	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

Callback Event Methods	Returns
onOffHook(*event*)	None
onHook(*event*)	None
onParking(*event*)	None
onPartyAdded(*event*)	None
onPartyChanged(*event*)	None
onPartyRemoved(*event*)	None
onReconnecting(*event*)	None
onReleasing(*event*)	None
onRetrieving(*event*)	None
onRinging(*event*)	None
onTalking(*event*)	None
onTransfering(*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

Fields

Field	Type
canAlternate	Boolean
canAnswer	Boolean
canAttachUserData	Boolean
canClear	Boolean
canComplete	Boolean

Field	Type
canConference	Boolean
canConferenceSingle	Boolean
canDropParty	Boolean
canHold	Boolean
canPark	Boolean
canReconnect	Boolean
canRelease	Boolean
canRetrievee	Boolean
canTransfer	Boolean
canTransferMute	Boolean
canUpdateCallData	Boolean

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, and callback event methods, and returns (if applicable) associated with this class.

Constructor

Constructor	Returns
MediaType()	MediaType object

Field

Field	Type
type	String of the following constants: • MT_EMAIL • MT_VOICE

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
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.

Constructor

Constructor	Returns
Reason(*code*, *desc*)	Reason object

Fields

Field	Type
code	String
description	String

RenServer

This section lists the constructor, fields, and 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 Methods	Returns
onDown(*event*)	None
onUp(*event*)	None

Server

This section lists the constructor, fields, methods, and 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: • ST_INSERVICE • ST_OUTOFSERVICE
type	String of the following constants: • TYPE_CTI • TYPE_UQ

Callback Event Methods

Callback Event Methods	Returns
onBroadcast(*event*)	None
onHbLost(*event*)	None
onHbRecovered(*event*)	None
onInService(*event*)	None
onOutOfService(*event*)	None
onRestart(*event*)	None

Session

This section lists the constructor, fields, methods, and 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
serverID	String
state	String of the following constants: • ST_ACTIVE • ST_CLOSED • ST_CLOSING • ST_IDLE
user	User object

Methods

Method	Returns
close()	None
open()	None
registerAddress(*address*)	None
registerBuddy(*buddy*)	None
registerGroup(*group*)	None
registerUser(*user*)	None
statPublish(*userStat*, *userStat*)	None
unregisterAddress(*address*)	None
unregisterBuddy(*buddy*)	None
unregisterGroup(*group*)	None
unregisterUser(*user*)	None

Callback Event Methods

Callback Event Methods	Returns
onAddressRegistered(*event*)	None
onAddressUnregistered(*event*)	None
onBuddyRegistered(*event*)	None
onBuddyUnregistered(*event*)	None
onClosed(*event*)	None
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

Field	Type
priority	String
type	String; one of the following: • TYPE_A2ACHAT • TYPE_CHAT • TYPE_CTI • TYPE_EMAIL • TYPE_GENERIC
urlAbs	Object
urlRel	Object

User

This section lists the constructor, fields, methods, and 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 286.

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 286.

Field	Type
addresses	Associative array
agentPassword	String
language	String
registeredAddresses	Associative array

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 Methods	Returns
onCapabilitiesChanged(*event*)	None
onCtiBusy(*event*)	None
onCtiClear(*event*)	None

Callback Event Methods	Returns
onDropped(*event*)	None
onError(*event*)	None
onInfo(*event*)	None
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

UserData

This section lists the constructor, fields, method, and returns (if applicable) associated with this class.

Constructor

Constructor	Returns
UserData()	UserData object

Constants

The UserData class uses the following constants:

- COMPONENT
- DESCR
- ISCRIPTPROGRAMNAME
- ICTYPE
- MARKET
- MENU
- REFERENCEID
- TARGET
- URLABS
- URLREL

Method

Method	Returns
addKeyValue(*key*, *value*)	None

UserStatistics1

This section lists the constructor, fields, and returns (if applicable) associated with this class.

Constructor

Constructor	Returns
UserStatistics1()	UserStatistics1 object

Fields

Field	Type
availableCost	String
ctiBusy	String

Field	Type
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, and callback event methods, and returns (if applicable) associated with this class.

Constructor

Constructor	Returns
UserStatistics2()	UserStatistics2 object

Fields

Field	Type
currentQueue	String
timeIdle	String
timeInCurrentState	String
timeNotReady	String
timeSinceLogin	String
totalTimeAvailable	String
totalTimeUnavailable	String

Glossary of PeopleSoft Terms

absence entitlement This element defines rules for granting paid time off for valid absences, such as sick time, vacation, and maternity leave. An absence entitlement element defines the entitlement amount, frequency, and entitlement period.

absence take This element defines the conditions that must be met before a payee is entitled to take paid time off.

academic career In PeopleSoft Enterprise Campus Solutions, all course work that a student undertakes at an academic institution and that is grouped in a single student record. For example, a university that has an undergraduate school, a graduate school, and various professional schools might define several academic careers—an undergraduate career, a graduate career, and separate careers for each professional school (law school, medical school, dental school, and so on).

academic institution In PeopleSoft Enterprise Campus Solutions, an entity (such as a university or college) that is independent of other similar entities and that has its own set of rules and business processes.

academic organization In PeopleSoft Enterprise Campus Solutions, an entity that is part of the administrative structure within an academic institution. At the lowest level, an academic organization might be an academic department. At the highest level, an academic organization can represent a division.

academic plan In PeopleSoft Enterprise Campus Solutions, an area of study—such as a major, minor, or specialization—that exists within an academic program or academic career.

academic program In PeopleSoft Enterprise Campus Solutions, the entity to which a student applies and is admitted and from which the student graduates.

accounting class In PeopleSoft Enterprise Performance Management, the accounting class defines how a resource is treated for generally accepted accounting practices. The Inventory class indicates whether a resource becomes part of a balance sheet account, such as inventory or fixed assets, while the Non-inventory class indicates that the resource is treated as an expense of the period during which it occurs.

accounting date The accounting date indicates when a transaction is recognized, as opposed to the date the transaction actually occurred. The accounting date and transaction date can be the same. The accounting date determines the period in the general ledger to which the transaction is to be posted. You can only select an accounting date that falls within an open period in the ledger to which you are posting. The accounting date for an item is normally the invoice date.

accounting split The accounting split method indicates how expenses are allocated or divided among one or more sets of accounting ChartFields.

accumulator You use an accumulator to store cumulative values of defined items as they are processed. You can accumulate a single value over time or multiple values over time. For example, an accumulator could consist of all voluntary deductions, or all company deductions, enabling you to accumulate amounts. It allows total flexibility for time periods and values accumulated.

action reason The reason an employee's job or employment information is updated. The action reason is entered in two parts: a personnel action, such as a promotion, termination, or change from one pay group to another—and a reason for that action. Action reasons are used by PeopleSoft Human Resources, PeopleSoft Benefits Administration,

PeopleSoft Stock Administration, and the COBRA Administration feature of the Base Benefits business process.

action template In PeopleSoft Receivables, outlines a set of escalating actions that the system or user performs based on the period of time that a customer or item has been in an action plan for a specific condition.

activity In PeopleSoft Enterprise Learning Management, an instance of a catalog item (sometimes called a class) that is available for enrollment. The activity defines such things as the costs that are associated with the offering, enrollment limits and deadlines, and waitlisting capacities.

In PeopleSoft Enterprise Performance Management, the work of an organization and the aggregation of actions that are used for activity-based costing.

In PeopleSoft Project Costing, the unit of work that provides a further breakdown of projects—usually into specific tasks.

In PeopleSoft Workflow, a specific transaction that you might need to perform in a business process. Because it consists of the steps that are used to perform a transaction, it is also known as a step map.

address usage In PeopleSoft Enterprise Campus Solutions, a grouping of address types defining the order in which the address types are used. For example, you might define an address usage code to process addresses in the following order: billing address, dormitory address, home address, and then work address.

adjustment calendar In PeopleSoft Enterprise Campus Solutions, the adjustment calendar controls how a particular charge is adjusted on a student's account when the student drops classes or withdraws from a term. The charge adjustment is based on how much time has elapsed from a predetermined date, and it is determined as a percentage of the original charge amount.

administrative function In PeopleSoft Enterprise Campus Solutions, a particular functional area that processes checklists, communication, and comments. The administrative function identifies which variable data is added to a person's checklist or communication record when a specific checklist code, communication category, or comment is assigned to the student. This key data enables you to trace that checklist, communication, or comment back to a specific processing event in a functional area.

admit type In PeopleSoft Enterprise Campus Solutions, a designation used to distinguish first-year applications from transfer applications.

agreement In PeopleSoft eSettlements, provides a way to group and specify processing options, such as payment terms, pay from a bank, and notifications by a buyer and supplier location combination.

allocation rule In PeopleSoft Enterprise Incentive Management, an expression within compensation plans that enables the system to assign transactions to nodes and participants. During transaction allocation, the allocation engine traverses the compensation structure from the current node to the root node, checking each node for plans that contain allocation rules.

alternate account A feature in PeopleSoft General Ledger that enables you to create a statutory chart of accounts and enter statutory account transactions at the detail transaction level, as required for recording and reporting by some national governments.

analysis database In PeopleSoft Enterprise Campus Solutions, database tables that store large amounts of student information that may not appear in standard report formats. The analysis database tables contain keys for all objects in a report that an application program can use to reference other student-record objects that are not contained in the printed report. For instance, the analysis database contains data on courses that are considered for satisfying a requirement but that are rejected. It also contains information on

courses captured by global limits. An analysis database is used in PeopleSoft Enterprise Academic Advisement.

Application Messaging PeopleSoft Application Messaging enables applications within the PeopleSoft Enterprise product family to communicate synchronously or asynchronously with other PeopleSoft and third-party applications. An application message defines the records and fields to be published or subscribed to.

AR specialist Abbreviation for *receivables specialist.* In PeopleSoft Receivables, an individual in who tracks and resolves deductions and disputed items.

arbitration plan In PeopleSoft Enterprise Pricer, defines how price rules are to be applied to the base price when the transaction is priced.

assessment rule In PeopleSoft Receivables, a user-defined rule that the system uses to evaluate the condition of a customer's account or of individual items to determine whether to generate a follow-up action.

asset class An asset group used for reporting purposes. It can be used in conjunction with the asset category to refine asset classification.

attribute/value pair In PeopleSoft Directory Interface, relates the data that makes up an entry in the directory information tree.

audience In PeopleSoft Enterprise Campus Solutions, a segment of the database that relates to an initiative, or a membership organization that is based on constituent attributes rather than a dues-paying structure. Examples of audiences include the Class of '65 and Undergraduate Arts & Sciences.

authentication server A server that is set up to verify users of the system.

base time period In PeopleSoft Business Planning, the lowest level time period in a calendar.

benchmark job In PeopleSoft Workforce Analytics, a benchmark job is a job code for which there is corresponding salary survey data from published, third-party sources.

billing career In PeopleSoft Enterprise Campus Solutions, the one career under which other careers are grouped for billing purposes if a student is active simultaneously in multiple careers.

bio bit or bio brief In PeopleSoft Enterprise Campus Solutions, a report that summarizes information stored in the system about a particular constituent. You can generate standard or specialized reports.

book In PeopleSoft Asset Management, used for storing financial and tax information, such as costs, depreciation attributes, and retirement information on assets.

branch A tree node that rolls up to nodes above it in the hierarchy, as defined in PeopleSoft Tree Manager.

budgetary account only An account used by the system only and not by users; this type of account does not accept transactions. You can only budget with this account. Formerly called "system-maintained account."

budget check In commitment control, the processing of source transactions against control budget ledgers, to see if they pass, fail, or pass with a warning.

budget control In commitment control, budget control ensures that commitments and expenditures don't exceed budgets. It enables you to track transactions against corresponding budgets and terminate a document's cycle if the defined budget conditions are not met. For example, you can prevent a purchase order from being dispatched to a vendor if there are insufficient funds in the related budget to support it.

budget period The interval of time (such as 12 months or 4 quarters) into which a period is divided for budgetary and reporting purposes. The ChartField allows maximum flexibility to define operational accounting time periods without restriction to only one calendar.

business activity The name of a subset of a detailed business process. This might be a specific transaction, task, or action that you perform in a business process.

business event In PeopleSoft Receivables, defines the processing characteristics for the Receivable Update process for a draft activity.

In PeopleSoft Sales Incentive Management, an original business transaction or activity that may justify the creation of a PeopleSoft Enterprise Incentive Management event (a sale, for example).

business process A standard set of 17 business processes are defined and maintained by the PeopleSoft product families and are supported by Business Process Engineering group at PeopleSoft. An example of a business process is Order Fulfillment, which is a business process that manages sales orders and contracts, inventory, billing, and so forth.

See also *detailed business process.*

business task The name of the specific function depicted in one of the business processes.

business unit A corporation or a subset of a corporation that is independent with regard to one or more operational or accounting functions.

buyer In PeopleSoft eSettlements, an organization (or business unit, as opposed to an individual) that transacts with suppliers (vendors) within the system. A buyer creates payments for purchases that are made in the system.

campus In PeopleSoft Enterprise Campus Solutions, an entity that is usually associated with a distinct physical administrative unit, that belongs to a single academic institution, that uses a unique course catalog, and that produces a common transcript for students within the same academic career.

catalog item In PeopleSoft Enterprise Learning Management, a specific topic that a learner can study and have tracked. For example, “Introduction to Microsoft Word.” A catalog item contains general information about the topic and includes a course code, description, categorization, keywords, and delivery methods. A catalog item can have one or more learning activities.

catalog map In PeopleSoft Catalog Management, translates values from the catalog source data to the format of the company’s catalog.

catalog partner In PeopleSoft Catalog Management, shares responsibility with the enterprise catalog manager for maintaining catalog content.

categorization Associates partner offerings with catalog offerings and groups them into enterprise catalog categories.

category In PeopleSoft Enterprise Campus Solutions, a broad grouping to which specific comments or communications (contexts) are assigned. Category codes are also linked to 3C access groups so that you can assign data-entry or view-only privileges across functions.

channel In PeopleSoft MultiChannel Framework, email, chat, voice (computer telephone integration [CTI]), or a generic event.

ChartField A field that stores a chart of accounts, resources, and so on, depending on the PeopleSoft application. ChartField values represent individual account numbers, department codes, and so forth.

ChartField balancing You can require specific ChartFields to match up (balance) on the debit and the credit side of a transaction.

ChartField combination edit The process of editing journal lines for valid ChartField combinations based on user-defined rules.

ChartKey One or more fields that uniquely identify each row in a table. Some tables contain only one field as the key, while others require a combination.

checkbook In PeopleSoft Promotions Management, enables you to view financial data (such as planned, incurred, and actual amounts) that is related to funds and trade promotions.

checklist code In PeopleSoft Enterprise Campus Solutions, a code that represents a list of planned or completed action items that can be assigned to a staff member, volunteer, or unit. Checklists enable you to view all action assignments on one page.

class In PeopleSoft Enterprise Campus Solutions, a specific offering of a course component within an academic term.

See also *course.*

Class ChartField A ChartField value that identifies a unique appropriation budget key when you combine it with a fund, department ID, and program code, as well as a budget period. Formerly called *sub-classification.*

clearance In PeopleSoft Enterprise Campus Solutions, the period of time during which a constituent in PeopleSoft Contributor Relations is approved for involvement in an initiative or an action. Clearances are used to prevent development officers from making multiple requests to a constituent during the same time period.

clone In PeopleCode, to make a unique copy. In contrast, to *copy* may mean making a new reference to an object, so if the underlying object is changed, both the copy and the original change.

cohort In PeopleSoft Enterprise Campus Solutions, the highest level of the three-level classification structure that you define for enrollment management. You can define a cohort level, link it to other levels, and set enrollment target numbers for it.

See also *population* and *division*.

collection To make a set of documents available for searching in Verity, you must first create at least one collection. A collection is set of directories and files that allow search application users to use the Verity search engine to quickly find and display source documents that match search criteria. A collection is a set of statistics and pointers to the source documents, stored in a proprietary format on a file server. Because a collection can only store information for a single location, PeopleSoft maintains a set of collections (one per language code) for each search index object.

collection rule In PeopleSoft Receivables, a user-defined rule that defines actions to take for a customer based on both the amount and the number of days past due for outstanding balances.

comm key See *communication key.*

communication key In PeopleSoft Enterprise Campus Solutions, a single code for entering a combination of communication category, communication context, communication method, communication direction, and standard letter code. Communication keys (also called *comm keys* or *speed keys*) can be created for background processes as well as for specific users.

compensation object In PeopleSoft Enterprise Incentive Management, a node within a compensation structure. Compensation objects are the building blocks that make up a compensation structure's hierarchical representation.

compensation structure In PeopleSoft Enterprise Incentive Management, a hierarchical relationship of compensation objects that represents the compensation-related relationship between the objects.

component interface A component interface is a set of application programming interfaces (APIs) that you can use to access and modify PeopleSoft database information using a program instead of the PeopleSoft client.

condition In PeopleSoft Receivables, occurs when there is a change of status for a customer's account, such as reaching a credit limit or exceeding a user-defined balance due.

configuration parameter catalog Used to configure an external system with PeopleSoft. For example, a configuration parameter catalog might set up configuration and communication parameters for an external server.

configuration plan In PeopleSoft Enterprise Incentive Management, configuration plans hold allocation information for common variables (not incentive rules) and are attached to a node without a participant. Configuration plans are not processed by transactions.

constituents In PeopleSoft Enterprise Campus Solutions, friends, alumni, organizations, foundations, or other entities affiliated with the institution, and about which the institution maintains information. The constituent types delivered with PeopleSoft Enterprise Contributor Relations Solutions are based on those defined by the Council for the Advancement and Support of Education (CASE).

content reference Content references are pointers to content registered in the portal registry. These are typically either URLs or iScripts. Content references fall into three categories: target content, templates, and template pagelets.

context In PeopleCode, determines which buffer fields can be contextually referenced and which is the current row of data on each scroll level when a PeopleCode program is running.

In PeopleSoft Enterprise Campus Solutions, a specific instance of a comment or communication. One or more contexts are assigned to a category, which you link to 3C access groups so that you can assign data-entry or view-only privileges across functions.

In PeopleSoft Enterprise Incentive Management, a mechanism that is used to determine the scope of a processing run. PeopleSoft Enterprise Incentive Management uses three types of context: plan, period, and run-level.

control table Stores information that controls the processing of an application. This type of processing might be consistent throughout an organization, or it might be used only by portions of the organization for more limited sharing of data.

cost profile A combination of a receipt cost method, a cost flow, and a deplete cost method. A profile is associated with a cost book and determines how items in that book are valued, as well as how the material movement of the item is valued for the book.

cost row A cost transaction and amount for a set of ChartFields.

course In PeopleSoft Enterprise Campus Solutions, a course that is offered by a school and that is typically described in a course catalog. A course has a standard syllabus and credit level; however, these may be modified at the class level. Courses can contain multiple components such as lecture, discussion, and lab.

See also *class*.

course share set In PeopleSoft Enterprise Campus Solutions, a tag that defines a set of requirement groups that can share courses. Course share sets are used in PeopleSoft Enterprise Academic Advisement.

current learning In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's in-progress learning activities and programs.

data acquisition In PeopleSoft Enterprise Incentive Management, the process during which raw business transactions are acquired from external source systems and fed into the operational data store (ODS).

data cube In PeopleSoft Analytic Calculation Engine, a data cube is a container for one kind of data (such as Sales data) and works with in tandem with one or more dimensions. Dimensions and data cubes in PeopleSoft Analytic Calculation Engine are unrelated to dimensions and online analytical processing (OLAP) cubes in PeopleSoft Cube Manager.

data elements Data elements, at their simplest level, define a subset of data and the rules by which to group them.

For Workforce Analytics, data elements are rules that tell the system what measures to retrieve about your workforce groups.

dataset A data grouping that enables role-based filtering and distribution of data. You can limit the range and quantity of data that is displayed for a user by associating dataset rules with user roles. The result of dataset rules is a set of data that is appropriate for the user's roles.

delivery method In PeopleSoft Enterprise Learning Management, identifies the primary type of delivery method in which a particular learning activity is offered. Also provides default values for the learning activity, such as cost and language. This is primarily used to help learners search the catalog for the type of delivery from which they learn best. Because PeopleSoft Enterprise Learning Management is a blended learning system, it does not enforce the delivery method.

In PeopleSoft Supply Chain Management, identifies the method by which goods are shipped to their destinations (such as truck, air, rail, and so on). The delivery method is specified when creating shipment schedules.

delivery method type In PeopleSoft Enterprise Learning Management, identifies how learning activities can be delivered—for example, through online learning, classroom instruction, seminars, books, and so forth—in an organization. The type determines whether the delivery method includes scheduled components.

detailed business process A subset of the business process. For example, the detailed business process named Determine Cash Position is a subset of the business process called Cash Management.

dimension In PeopleSoft Analytic Calculation Engine, a dimension contains a list of one kind of data that can span various contexts, and it is a basic component of an analytic model. Within the analytic model, a dimension is attached to one or more data cubes. In PeopleSoft Cube Manager, a dimension is the most basic component of an OLAP cube and specifies the PeopleSoft metadata to be used to create the dimension's rollup structure. Dimensions and data cubes in PeopleSoft Analytic Calculation Engine are unrelated to dimensions and OLAP cubes in PeopleSoft Cube Manager.

directory information tree In PeopleSoft Directory Interface, the representation of a directory's hierarchical structure.

division In PeopleSoft Enterprise Campus Solutions, the lowest level of the three-level classification structure that you define in PeopleSoft Enterprise Recruiting and Admissions for enrollment management. You can define a division level, link it to other levels, and set enrollment target numbers for it.

See also *population* and *cohort*.

document sequencing A flexible method that sequentially numbers the financial transactions (for example, bills, purchase orders, invoices, and payments) in the system for statutory reporting and for tracking commercial transaction activity.

dynamic detail tree A tree that takes its detail values—dynamic details—directly from a table in the database, rather than from a range of values that are entered by the user.

edit table A table in the database that has its own record definition, such as the Department table. As fields are entered into a PeopleSoft application, they can be validated against an edit table to ensure data integrity throughout the system.

effective date A method of dating information in PeopleSoft applications. You can predate information to add historical data to your system, or postdate information in order to enter it before it actually goes into effect. By using effective dates, you don't delete values; you enter a new value with a current effective date.

EIM ledger Abbreviation for *Enterprise Incentive Management ledger.* In PeopleSoft Enterprise Incentive Management, an object to handle incremental result gathering within the scope of a participant. The ledger captures a result set with all of the appropriate traces to the data origin and to the processing steps of which it is a result.

elimination set In PeopleSoft General Ledger, a related group of intercompany accounts that is processed during consolidations.

entry event In PeopleSoft General Ledger, Receivables, Payables, Purchasing, and Billing, a business process that generates multiple debits and credits resulting from single transactions to produce standard, supplemental accounting entries.

equitization In PeopleSoft General Ledger, a business process that enables parent companies to calculate the net income of subsidiaries on a monthly basis and adjust that amount to increase the investment amount and equity income amount before performing consolidations.

equity item limit In PeopleSoft Enterprise Campus Solutions, the amounts of funds set by the institution to be awarded with discretionary or gift funds. The limit could be reduced by amounts equal to such things as expected family contribution (EFC) or parent contribution. Students are packaged by Equity Item Type Groups and Related Equity Item Types. This limit can be used to assure that similar student populations are packaged equally.

event A predefined point either in the Component Processor flow or in the program flow. As each point is encountered, the event activates each component, triggering any PeopleCode program that is associated with that component and that event. Examples of events are FieldChange, SavePreChange, and RowDelete.

In PeopleSoft Human Resources, also refers to an incident that affects benefits eligibility.

event propagation process In PeopleSoft Sales Incentive Management, a process that determines, through logic, the propagation of an original PeopleSoft Enterprise Incentive Management event and creates a derivative (duplicate) of the original event to be processed by other objects. Sales Incentive Management uses this mechanism to implement splits, roll-ups, and so on. Event propagation determines who receives the credit.

exception In PeopleSoft Receivables, an item that either is a deduction or is in dispute.

exclusive pricing In PeopleSoft Order Management, a type of arbitration plan that is associated with a price rule. Exclusive pricing is used to price sales order transactions.

fact In PeopleSoft applications, facts are numeric data values from fields from a source database as well as an analytic application. A fact can be anything you want to measure your business by, for example, revenue, actual, budget data, or sales numbers. A fact is stored on a fact table.

financial aid term In PeopleSoft Enterprise Campus Solutions, a combination of a period of time that the school determines as an instructional accounting period and an academic career. It is created and defined during the setup process. Only terms eligible for financial aid are set up for each financial aid career.

forecast item A logical entity with a unique set of descriptive demand and forecast data that is used as the basis to forecast demand. You create forecast items for a wide range of uses, but they ultimately represent things that you buy, sell, or use in your organization and for which you require a predictable usage.

fund In PeopleSoft Promotions Management, a budget that can be used to fund promotional activity. There are four funding methods: top down, fixed accrual, rolling accrual, and zero-based accrual.

gap In PeopleSoft Enterprise Campus Solutions, an artificial figure that sets aside an amount of unmet financial aid need that is not funded with Title IV funds. A gap can be used to prevent fully funding any student to conserve funds, or it can be used to preserve unmet financial aid need so that institutional funds can be awarded.

generic process type In PeopleSoft Process Scheduler, process types are identified by a generic process type. For example, the generic process type SQR includes all SQR process types, such as SQR process and SQR report.

gift table In PeopleSoft Enterprise Campus Solutions, a table or so-called *donor pyramid* describing the number and size of gifts that you expect will be needed to successfully complete the campaign in PeopleSoft Contributor Relations. The gift table enables you to estimate the number of donors and prospects that you need at each gift level to reach the campaign goal.

GL business unit Abbreviation for *general ledger business unit.* A unit in an organization that is an independent entity for accounting purposes. It maintains its own set of accounting books.

See also *business unit*.

GL entry template Abbreviation for *general ledger entry template.* In PeopleSoft Enterprise Campus Solutions, a template that defines how a particular item is sent to the general ledger. An item-type maps to the general ledger, and the GL entry template can involve multiple general ledger accounts. The entry to the general ledger is further controlled by high-level flags that control the summarization and the type of accounting—that is, accrual or cash.

GL Interface process Abbreviation for *General Ledger Interface process.* In PeopleSoft Enterprise Campus Solutions, a process that is used to send transactions from PeopleSoft Enterprise Student Financials to the general ledger. Item types are mapped to specific general ledger accounts, enabling transactions to move to the general ledger when the GL Interface process is run.

group In PeopleSoft Billing and Receivables, a posting entity that comprises one or more transactions (items, deposits, payments, transfers, matches, or write-offs).

In PeopleSoft Human Resources Management and Supply Chain Management, any set of records that are associated under a single name or variable to run calculations in PeopleSoft business processes. In PeopleSoft Time and Labor, for example, employees are placed in groups for time reporting purposes.

incentive object In PeopleSoft Enterprise Incentive Management, the incentive-related objects that define and support the PeopleSoft Enterprise Incentive Management calculation process and results, such as plan templates, plans, results data, user interaction objects, and so on.

incentive rule In PeopleSoft Sales Incentive Management, the commands that act on transactions and turn them into compensation. A rule is one part in the process of turning a transaction into compensation.

incur In PeopleSoft Promotions Management, to become liable for a promotional payment. In other words, you owe that amount to a customer for promotional activities.

initiative In PeopleSoft Enterprise Campus Solutions, the basis from which all advancement plans are executed. It is an organized effort targeting a specific constituency, and it can occur over a specified period of time with specific purposes and goals. An initiative can be a campaign, an event, an organized volunteer effort, a membership drive, or any other type of effort defined by the institution. Initiatives can be multipart, and they can be related to other initiatives. This enables you to track individual parts of an initiative, as well as entire initiatives.

inquiry access In PeopleSoft Enterprise Campus Solutions, a type of security access that permits the user only to view data.

See also *update access.*

institution In PeopleSoft Enterprise Campus Solutions, an entity (such as a university or college) that is independent of other similar entities and that has its own set of rules and business processes.

integration A relationship between two compatible integration points that enables communication to take place between systems. Integrations enable PeopleSoft applications to work seamlessly with other PeopleSoft applications or with third-party systems or software.

integration point An interface that a system uses to communicate with another PeopleSoft application or an external application.

integration set A logical grouping of integrations that applications use for the same business purpose. For example, the integration set ADVANCED_SHIPPING_ORDER contains all of the integrations that notify a customer that an order has shipped.

item In PeopleSoft Inventory, a tangible commodity that is stored in a business unit (shipped from a warehouse).

In PeopleSoft Demand Planning, Inventory Policy Planning, and Supply Planning, a noninventory item that is designated as being used for planning purposes only. It can represent a family or group of inventory items. It can have a planning bill of material (BOM) or planning routing, and it can exist as a component on a planning BOM. A planning item cannot be specified on a production or engineering BOM or routing, and it cannot be used as a component in a production. The quantity on hand will never be maintained.

In PeopleSoft Receivables, an individual receivable. An item can be an invoice, a credit memo, a debit memo, a write-off, or an adjustment.

item shuffle In PeopleSoft Enterprise Campus Solutions, a process that enables you to change a payment allocation without having to reverse the payment.

joint communication In PeopleSoft Enterprise Campus Solutions, one letter that is addressed jointly to two people. For example, a letter might be addressed to both Mr. Sudhir Awat and Ms. Samantha Mortelli. A relationship must be established between the two individuals in the database, and at least one of the individuals must have an ID in the database.

keyword In PeopleSoft Enterprise Campus Solutions, a term that you link to particular elements within PeopleSoft Student Financials, Financial Aid, and Contributor Relations. You can use keywords as search criteria that enable you to locate specific records in a search dialog box.

KPI An abbreviation for *key performance indicator.* A high-level measurement of how well an organization is doing in achieving critical success factors. This defines the data value or calculation upon which an assessment is determined.

LDIF file Abbreviation for *Lightweight Directory Access Protocol (LDAP) Data Interchange Format file.* Contains discrepancies between PeopleSoft data and directory data.

learner group In PeopleSoft Enterprise Learning Management, a group of learners who are linked to the same learning environment. Members of the learner group can share the same attributes, such as the same department or job code. Learner groups are used to control access to and enrollment in learning activities and programs. They are also used to perform group enrollments and mass enrollments in the back office.

learning components In PeopleSoft Enterprise Learning Management, the foundational building blocks of learning activities. PeopleSoft Enterprise Learning Management supports six basic types of learning components: web-based, session, webcast, test, survey, and assignment. One or more of these learning component types compose a single learning activity.

learning environment In PeopleSoft Enterprise Learning Management, identifies a set of categories and catalog items that can be made available to learner groups. Also defines the default values that are assigned to the learning activities and programs that are created within a particular learning environment. Learning environments provide a way to partition the catalog so that learners see only those items that are relevant to them.

learning history In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's completed learning activities and programs.

ledger mapping You use ledger mapping to relate expense data from general ledger accounts to resource objects. Multiple ledger line items can be mapped to one or more resource IDs. You can also use ledger mapping to map dollar amounts (referred to as *rates*) to business units. You can map the amounts in two different ways: an actual amount that represents actual costs of the accounting period, or a budgeted amount that can be used to calculate the capacity rates as well as budgeted model results. In PeopleSoft Enterprise Warehouse, you can map general ledger accounts to the EW Ledger table.

library section In PeopleSoft Enterprise Incentive Management, a section that is defined in a plan (or template) and that is available for other plans to share. Changes to a library section are reflected in all plans that use it.

linked section In PeopleSoft Enterprise Incentive Management, a section that is defined in a plan template but appears in a plan. Changes to linked sections propagate to plans using that section.

linked variable In PeopleSoft Enterprise Incentive Management, a variable that is defined and maintained in a plan template and that also appears in a plan. Changes to linked variables propagate to plans using that variable.

LMS Abbreviation for *learning management system.* In PeopleSoft Enterprise Campus Solutions, LMS is a PeopleSoft Student Records feature that provides a common set of interoperability standards that enable the sharing of instructional content and data between learning and administrative environments.

load In PeopleSoft Inventory, identifies a group of goods that are shipped together. Load management is a feature of PeopleSoft Inventory that is used to track the weight, the volume, and the destination of a shipment.

local functionality In PeopleSoft HRMS, the set of information that is available for a specific country. You can access this information when you click the appropriate country flag in the global window, or when you access it by a local country menu.

location Locations enable you to indicate the different types of addresses—for a company, for example, one address to receive bills, another for shipping, a third for postal deliveries, and a separate street address. Each address has a different location number. The primary location—indicated by a *1*—is the address you use most often and may be different from the main address.

logistical task In PeopleSoft Services Procurement, an administrative task that is related to hiring a service provider. Logistical tasks are linked to the service type on the work order so that different types of services can have different logistical tasks. Logistical tasks include both preapproval tasks (such as assigning a new badge or ordering a new laptop) and postapproval tasks (such as scheduling orientation or setting up the service provider email). The logistical tasks can be mandatory or optional. Mandatory preapproval tasks must be completed before the work order is approved. Mandatory postapproval tasks, on the other hand, must be completed before a work order is released to a service provider.

market template In PeopleSoft Enterprise Incentive Management, additional functionality that is specific to a given market or industry and is built on top of a product category.

mass change In PeopleSoft Enterprise Campus Solutions, mass change is a SQL generator that can be used to create specialized functionality. Using mass change, you can set up a series of Insert, Update, or Delete SQL statements to perform business functions that are specific to the institution.

See also *3C engine*.

match group In PeopleSoft Receivables, a group of receivables items and matching offset items. The system creates match groups by using user-defined matching criteria for selected field values.

MCF server Abbreviation for *PeopleSoft MultiChannel Framework server.* Comprises the universal queue server and the MCF log server. Both processes are started when *MCF Servers* is selected in an application server domain configuration.

merchandising activity In PeopleSoft Promotions Management, a specific discount type that is associated with a trade promotion (such as off-invoice, billback or rebate, or lump-sum payment) that defines the performance that is required to receive the discount. In the industry, you may know this as an offer, a discount, a merchandising event, an event, or a tactic.

meta-SQL Meta-SQL constructs expand into platform-specific Structured Query Language (SQL) substrings. They are used in functions that pass SQL strings, such as in SQL objects, the SQLExec function, and PeopleSoft Application Engine programs.

metastring Metastrings are special expressions included in SQL string literals. The metastrings, prefixed with a percent (%) symbol, are included directly in the string literals. They expand at run time into an appropriate substring for the current database platform.

multibook In PeopleSoft General Ledger, multiple ledgers having multiple-base currencies that are defined for a business unit, with the option to post a single transaction to all base currencies (all ledgers) or to only one of those base currencies (ledgers).

multicurrency The ability to process transactions in a currency other than the business unit's base currency.

national allowance In PeopleSoft Promotions Management, a promotion at the corporate level that is funded by nondiscretionary dollars. In the industry, you may know this as a national promotion, a corporate promotion, or a corporate discount.

need In PeopleSoft Enterprise Campus Solutions, the difference between the cost of attendance (COA) and the expected family contribution (EFC). It is the gap between the cost of attending the school and the student's resources. The financial aid package

	is based on the amount of financial need. The process of determining a student's need is called *need analysis.*
node-oriented tree	A tree that is based on a detail structure, but the detail values are not used.
pagelet	Each block of content on the home page is called a pagelet. These pagelets display summary information within a small rectangular area on the page. The pagelet provide users with a snapshot of their most relevant PeopleSoft and non-PeopleSoft content.
participant	In PeopleSoft Enterprise Incentive Management, participants are recipients of the incentive compensation calculation process.
participant object	Each participant object may be related to one or more compensation objects. See also *compensation object.*
partner	A company that supplies products or services that are resold or purchased by the enterprise.
pay cycle	In PeopleSoft Payables, a set of rules that define the criteria by which it should select scheduled payments for payment creation.
payment shuffle	In PeopleSoft Enterprise Campus Solutions, a process allowing payments that have been previously posted to a student's account to be automatically reapplied when a higher priority payment is posted or the payment allocation definition is changed.
pending item	In PeopleSoft Receivables, an individual receivable (such as an invoice, a credit memo, or a write-off) that has been entered in or created by the system, but hasn't been posted.
PeopleCode	PeopleCode is a proprietary language, executed by the PeopleSoft component processor. PeopleCode generates results based on existing data or user actions. By using various tools provided with PeopleTools, external services are available to all PeopleSoft applications wherever PeopleCode can be executed.
PeopleCode event	See *event.*
PeopleSoft Pure Internet Architecture	The fundamental architecture on which PeopleSoft 8 applications are constructed, consisting of a relational database management system (RDBMS), an application server, a web server, and a browser.
performance measurement	In PeopleSoft Enterprise Incentive Management, a variable used to store data (similar to an aggregator, but without a predefined formula) within the scope of an incentive plan. Performance measures are associated with a plan calendar, territory, and participant. Performance measurements are used for quota calculation and reporting.
period context	In PeopleSoft Enterprise Incentive Management, because a participant typically uses the same compensation plan for multiple periods, the period context associates a plan context with a specific calendar period and fiscal year. The period context references the associated plan context, thus forming a chain. Each plan context has a corresponding set of period contexts.
person of interest	A person about whom the organization maintains information but who is not part of the workforce.
personal portfolio	In PeopleSoft Enterprise Campus Solutions, the user-accessible menu item that contains an individual's name, address, telephone number, and other personal information.
plan	In PeopleSoft Sales Incentive Management, a collection of allocation rules, variables, steps, sections, and incentive rules that instruct the PeopleSoft Enterprise Incentive Management engine in how to process transactions.

plan context In PeopleSoft Enterprise Incentive Management, correlates a participant with the compensation plan and node to which the participant is assigned, enabling the PeopleSoft Enterprise Incentive Management system to find anything that is associated with the node and that is required to perform compensation processing. Each participant, node, and plan combination represents a unique plan context—if three participants are on a compensation structure, each has a different plan context. Configuration plans are identified by plan contexts and are associated with the participants that refer to them.

plan template In PeopleSoft Enterprise Incentive Management, the base from which a plan is created. A plan template contains common sections and variables that are inherited by all plans that are created from the template. A template may contain steps and sections that are not visible in the plan definition.

planned learning In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's planned learning activities and programs.

planning instance In PeopleSoft Supply Planning, a set of data (business units, items, supplies, and demands) constituting the inputs and outputs of a supply plan.

population In PeopleSoft Enterprise Campus Solutions, the middle level of the three-level classification structure that you define in PeopleSoft Enterprise Recruiting and Admissions for enrollment management. You can define a population level, link it to other levels, and set enrollment target numbers for it.

See also *division* and *cohort*.

portal registry In PeopleSoft applications, the portal registry is a tree-like structure in which content references are organized, classified, and registered. It is a central repository that defines both the structure and content of a portal through a hierarchical, tree-like structure of folders useful for organizing and securing content references.

price list In PeopleSoft Enterprise Pricer, enables you to select products and conditions for which the price list applies to a transaction. During a transaction, the system either determines the product price based on the predefined search hierarchy for the transaction or uses the product's lowest price on any associated, active price lists. This price is used as the basis for any further discounts and surcharges.

price rule In PeopleSoft Enterprise Pricer, defines the conditions that must be met for adjustments to be applied to the base price. Multiple rules can apply when conditions of each rule are met.

price rule condition In PeopleSoft Enterprise Pricer, selects the price-by fields, the values for the price-by fields, and the operator that determines how the price-by fields are related to the transaction.

price rule key In PeopleSoft Enterprise Pricer, defines the fields that are available to define price rule conditions (which are used to match a transaction) on the price rule.

primacy number In PeopleSoft Enterprise Campus Solutions, a number that the system uses to prioritize financial aid applications when students are enrolled in multiple academic careers and academic programs at the same time. The Consolidate Academic Statistics process uses the primacy number indicated for both the career and program at the institutional level to determine a student's primary career and program. The system also uses the number to determine the primary student attribute value that is used when you extract data to report on cohorts. The lowest number takes precedence.

primary name type In PeopleSoft Enterprise Campus Solutions, the name type that is used to link the name stored at the highest level within the system to the lower-level set of names that an individual provides.

process category In PeopleSoft Process Scheduler, processes that are grouped for server load balancing and prioritization.

process group In PeopleSoft Financials, a group of application processes (performed in a defined order) that users can initiate in real time, directly from a transaction entry page.

process definition Process definitions define each run request.

process instance A unique number that identifies each process request. This value is automatically incremented and assigned to each requested process when the process is submitted to run.

process job You can link process definitions into a job request and process each request serially or in parallel. You can also initiate subsequent processes based on the return code from each prior request.

process request A single run request, such as a Structured Query Report (SQR), a COBOL or Application Engine program, or a Crystal report that you run through PeopleSoft Process Scheduler.

process run control A PeopleTools variable used to retain PeopleSoft Process Scheduler values needed at runtime for all requests that reference a run control ID. Do not confuse these with application run controls, which may be defined with the same run control ID, but only contain information specific to a given application process request.

product A PeopleSoft or third-party product. PeopleSoft organizes its software products into product families and product lines. Interactive Services Repository contains information about every release of every product that PeopleSoft sells, as well as products from certified third-party companies. These products are displayed with the product name and release number.

product category In PeopleSoft Enterprise Incentive Management, indicates an application in the Enterprise Incentive Management suite of products. Each transaction in the PeopleSoft Enterprise Incentive Management system is associated with a product category.

product family A group of products that are related by common functionality. The family names that can be searched using Interactive Service Repository are PeopleSoft Enterprise, PeopleSoft EnterpriseOne, PeopleSoft World, and third-party, certified PeopleSoft partners.

product line The name of a PeopleSoft product line or the company name of a third-party certified partner. Integration Services Repository enables you to search for integration points by product line.

programs In PeopleSoft Enterprise Learning Management, a high-level grouping that guides the learner along a specific learning path through sections of catalog items. PeopleSoft Enterprise Learning Systems provides two types of programs—curricula and certifications.

progress log In PeopleSoft Services Procurement, tracks deliverable-based projects. This is similar to the time sheet in function and process. The service provider contact uses the progress log to record and submit progress on deliverables. The progress can be logged by the activity that is performed, by the percentage of work that is completed, or by the completion of milestone activities that are defined for the project.

project transaction In PeopleSoft Project Costing, an individual transaction line that represents a cost, time, budget, or other transaction row.

promotion In PeopleSoft Promotions Management, a trade promotion, which is typically funded from trade dollars and used by consumer products manufacturers to increase sales volume.

prospects

In PeopleSoft Enterprise Campus Solutions, students who are interested in applying to the institution.

In PeopleSoft Enterprise Contributor Relations, individuals and organizations that are most likely to make substantial financial commitments or other types of commitments to the institution.

publishing

In PeopleSoft Enterprise Incentive Management, a stage in processing that makes incentive-related results available to participants.

rating components

In PeopleSoft Enterprise Campus Solutions, variables used with the Equation Editor to retrieve specified populations.

record group

A set of logically and functionally related control tables and views. Record groups help enable TableSet sharing, which eliminates redundant data entry. Record groups ensure that TableSet sharing is applied consistently across all related tables and views.

record input VAT flag

Abbreviation for *record input value-added tax flag.* Within PeopleSoft Purchasing, Payables, and General Ledger, this flag indicates that you are recording input VAT on the transaction. This flag, in conjunction with the record output VAT flag, is used to determine the accounting entries created for a transaction and to determine how a transaction is reported on the VAT return. For all cases within Purchasing and Payables where VAT information is tracked on a transaction, this flag is set to Yes. This flag is not used in PeopleSoft Order Management, Billing, or Receivables, where it is assumed that you are always recording only output VAT, or in PeopleSoft Expenses, where it is assumed that you are always recording only input VAT.

record output VAT flag

Abbreviation for *record output value-added tax flag.*

See *record input VAT flag.*

recname

The name of a record that is used to determine the associated field to match a value or set of values.

recognition

In PeopleSoft Enterprise Campus Solutions, the recognition type indicates whether the PeopleSoft Enterprise Contributor Relations donor is the primary donor of a commitment or shares the credit for a donation. Primary donors receive hard credit that must total 100 percent. Donors that share the credit are given soft credit. Institutions can also define other share recognition-type values such as memo credit or vehicle credit.

reference data

In PeopleSoft Sales Incentive Management, system objects that represent the sales organization, such as territories, participants, products, customers, channels, and so on.

reference object

In PeopleSoft Enterprise Incentive Management, this dimension-type object further defines the business. Reference objects can have their own hierarchy (for example, product tree, customer tree, industry tree, and geography tree).

reference transaction

In commitment control, a reference transaction is a source transaction that is referenced by a higher-level (and usually later) source transaction, in order to automatically reverse all or part of the referenced transaction's budget-checked amount. This avoids duplicate postings during the sequential entry of the transaction at different commitment levels. For example, the amount of an encumbrance transaction (such as a purchase order) will, when checked and recorded against a budget, cause the system to concurrently reference and relieve all or part of the amount of a corresponding pre-encumbrance transaction, such as a purchase requisition.

regional sourcing

In PeopleSoft Purchasing, provides the infrastructure to maintain, display, and select an appropriate vendor and vendor pricing structure that is based on a regional sourcing model where the multiple ship to locations are grouped. Sourcing may occur at a level higher than the ship to location.

relationship object In PeopleSoft Enterprise Incentive Management, these objects further define a compensation structure to resolve transactions by establishing associations between compensation objects and business objects.

remote data source data Data that is extracted from a separate database and migrated into the local database.

REN server Abbreviation for *real-time event notification server* in PeopleSoft MultiChannel Framework.

requester In PeopleSoft eSettlements, an individual who requests goods or services and whose ID appears on the various procurement pages that reference purchase orders.

reversal indicator In PeopleSoft Enterprise Campus Solutions, an indicator that denotes when a particular payment has been reversed, usually because of insufficient funds.

role Describes how people fit into PeopleSoft Workflow. A role is a class of users who perform the same type of work, such as clerks or managers. Your business rules typically specify what user role needs to do an activity.

role user A PeopleSoft Workflow user. A person's role user ID serves much the same purpose as a user ID does in other parts of the system. PeopleSoft Workflow uses role user IDs to determine how to route worklist items to users (through an email address, for example) and to track the roles that users play in the workflow. Role users do not need PeopleSoft user IDs.

roll up In a tree, to roll up is to total sums based on the information hierarchy.

run control A run control is a type of online page that is used to begin a process, such as the batch processing of a payroll run. Run control pages generally start a program that manipulates data.

run control ID A unique ID to associate each user with his or her own run control table entries.

run-level context In PeopleSoft Enterprise Incentive Management, associates a particular run (and batch ID) with a period context and plan context. Every plan context that participates in a run has a separate run-level context. Because a run cannot span periods, only one run-level context is associated with each plan context.

SCP SCBM XML message Abbreviation for *Supply Chain Planning Supply Chain Business Modeler Extensible Markup Language message.* PeopleSoft EnterpriseOne Supply Chain Business Modeler uses XML as the format for all data that it imports and exports.

search query You use this set of objects to pass a query string and operators to the search engine. The search index returns a set of matching results with keys to the source documents.

search/match In PeopleSoft Enterprise Campus Solutions and PeopleSoft Enterprise Human Resources Management Solutions, a feature that enables you to search for and identify duplicate records in the database.

seasonal address In PeopleSoft Enterprise Campus Solutions, an address that recurs for the same length of time at the same time of year each year until adjusted or deleted.

section In PeopleSoft Enterprise Incentive Management, a collection of incentive rules that operate on transactions of a specific type. Sections enable plans to be segmented to process logical events in different sections.

security event In commitment control, security events trigger security authorization checking, such as budget entries, transfers, and adjustments; exception overrides and notifications; and inquiries.

serial genealogy In PeopleSoft Manufacturing, the ability to track the composition of a specific, serial-controlled item.

serial in production In PeopleSoft Manufacturing, enables the tracing of serial information for manufactured items. This is maintained in the Item Master record.

service impact In PeopleSoft Enterprise Campus Solutions, the resulting action triggered by a service indicator. For example, a service indicator that reflects nonpayment of account balances by a student might result in a service impact that prohibits registration for classes.

service indicator In PeopleSoft Enterprise Campus Solutions, indicates services that may be either withheld or provided to an individual. Negative service indicators indicate holds that prevent the individual from receiving specified services, such as check-cashing privileges or registration for classes. Positive service indicators designate special services that are provided to the individual, such as front-of-line service or special services for disabled students.

session In PeopleSoft Enterprise Campus Solutions, time elements that subdivide a term into multiple time periods during which classes are offered. In PeopleSoft Contributor Relations, a session is the means of validating gift, pledge, membership, or adjustment data entry . It controls access to the data entered by a specific user ID. Sessions are balanced, queued, and then posted to the institution's financial system. Sessions must be posted to enter a matching gift or pledge payment, to make an adjustment, or to process giving clubs or acknowledgements.

In PeopleSoft Enterprise Learning Management, a single meeting day of an activity (that is, the period of time between start and finish times within a day). The session stores the specific date, location, meeting time, and instructor. Sessions are used for scheduled training.

session template In PeopleSoft Enterprise Learning Management, enables you to set up common activity characteristics that may be reused while scheduling a PeopleSoft Enterprise Learning Management activity—characteristics such as days of the week, start and end times, facility and room assignments, instructors, and equipment. A session pattern template can be attached to an activity that is being scheduled. Attaching a template to an activity causes all of the default template information to populate the activity session pattern.

setup relationship In PeopleSoft Enterprise Incentive Management, a relationship object type that associates a configuration plan with any structure node.

share driver expression In PeopleSoft Business Planning, a named planning method similar to a driver expression, but which you can set up globally for shared use within a single planning application or to be shared between multiple planning applications through PeopleSoft Enterprise Warehouse.

single signon With single signon, users can, after being authenticated by a PeopleSoft application server, access a second PeopleSoft application server without entering a user ID or password.

source key process In PeopleSoft Enterprise Campus Solutions, a process that relates a particular transaction to the source of the charge or financial aid. On selected pages, you can drill down into particular charges.

source transaction In commitment control, any transaction generated in a PeopleSoft or third-party application that is integrated with commitment control and which can be checked against commitment control budgets. For example, a pre-encumbrance, encumbrance, expenditure, recognized revenue, or collected revenue transaction.

speed key See *communication key.*

SpeedChart A user-defined shorthand key that designates several ChartKeys to be used for voucher entry. Percentages can optionally be related to each ChartKey in a SpeedChart definition.

SpeedType A code representing a combination of ChartField values. SpeedTypes simplify the entry of ChartFields commonly used together.

staging A method of consolidating selected partner offerings with the offerings from the enterprise's other partners.

standard letter code In PeopleSoft Enterprise Campus Solutions, a standard letter code used to identify each letter template available for use in mail merge functions. Every letter generated in the system must have a standard letter code identification.

statutory account Account required by a regulatory authority for recording and reporting financial results. In PeopleSoft, this is equivalent to the Alternate Account (ALTACCT) ChartField.

step In PeopleSoft Sales Incentive Management, a collection of sections in a plan. Each step corresponds to a step in the job run.

storage level In PeopleSoft Inventory, identifies the level of a material storage location. Material storage locations are made up of a business unit, a storage area, and a storage level. You can set up to four storage levels.

subcustomer qualifier A value that groups customers into a division for which you can generate detailed history, aging, events, and profiles.

Summary ChartField You use summary ChartFields to create summary ledgers that roll up detail amounts based on specific detail values or on selected tree nodes. When detail values are summarized using tree nodes, summary ChartFields must be used in the summary ledger data record to accommodate the maximum length of a node name (20 characters).

summary ledger An accounting feature used primarily in allocations, inquiries, and PS/nVision reporting to store combined account balances from detail ledgers. Summary ledgers increase speed and efficiency of reporting by eliminating the need to summarize detail ledger balances each time a report is requested. Instead, detail balances are summarized in a background process according to user-specified criteria and stored on summary ledgers. The summary ledgers are then accessed directly for reporting.

summary time period In PeopleSoft Business Planning, any time period (other than a base time period) that is an aggregate of other time periods, including other summary time periods and base time periods, such as quarter and year total.

summary tree A tree used to roll up accounts for each type of report in summary ledgers. Summary trees enable you to define trees on trees. In a summary tree, the detail values are really nodes on a detail tree or another summary tree (known as the *basis* tree). A summary tree structure specifies the details on which the summary trees are to be built.

syndicate To distribute a production version of the enterprise catalog to partners.

system function In PeopleSoft Receivables, an activity that defines how the system generates accounting entries for the general ledger.

TableSet A means of sharing similar sets of values in control tables, where the actual data values are different but the structure of the tables is the same.

TableSet sharing Shared data that is stored in many tables that are based on the same TableSets. Tables that use TableSet sharing contain the SETID field as an additional key or unique identifier.

target currency The value of the entry currency or currencies converted to a single currency for budget viewing and inquiry purposes.

tax authority In PeopleSoft Enterprise Campus Solutions, a user-defined element that combines a description and percentage of a tax with an account type, an item type, and a service impact.

template A template is HTML code associated with a web page. It defines the layout of the page and also where to get HTML for each part of the page. In PeopleSoft, you use templates to build a page by combining HTML from a number of sources. For a PeopleSoft portal, all templates must be registered in the portal registry, and each content reference must be assigned a template.

territory In PeopleSoft Sales Incentive Management, hierarchical relationships of business objects, including regions, products, customers, industries, and participants.

third party A company or vendor that has extensive PeopleSoft product knowledge and whose products and integrations have been certified and are compatible with PeopleSoft applications.

3C engine Abbreviation for *Communications, Checklists, and Comments engine.* In PeopleSoft Enterprise Campus Solutions, the 3C engine enables you to automate business processes that involve additions, deletions, and updates to communications, checklists, and comments. You define events and triggers to engage the engine, which runs the mass change and processes the 3C records (for individuals or organizations) immediately and automatically from within business processes.

3C group Abbreviation for *Communications, Checklists, and Comments group.* In PeopleSoft Enterprise Campus Solutions, a method of assigning or restricting access privileges. A 3C group enables you to group specific communication categories, checklist codes, and comment categories. You can then assign the group inquiry-only access or update access, as appropriate.

TimeSpan A relative period, such as year-to-date or current period, that can be used in various PeopleSoft General Ledger functions and reports when a rolling time frame, rather than a specific date, is required. TimeSpans can also be used with flexible formulas in PeopleSoft Projects.

trace usage In PeopleSoft Manufacturing, enables the control of which components will be traced during the manufacturing process. Serial- and lot-controlled components can be traced. This is maintained in the Item Master record.

transaction allocation In PeopleSoft Enterprise Incentive Management, the process of identifying the owner of a transaction. When a raw transaction from a batch is allocated to a plan context, the transaction is duplicated in the PeopleSoft Enterprise Incentive Management transaction tables.

transaction state In PeopleSoft Enterprise Incentive Management, a value assigned by an incentive rule to a transaction. Transaction states enable sections to process only transactions that are at a specific stage in system processing. After being successfully processed, transactions may be promoted to the next transaction state and "picked up" by a different section for further processing.

Translate table A system edit table that stores codes and translate values for the miscellaneous fields in the database that do not warrant individual edit tables of their own.

tree The graphical hierarchy in PeopleSoft systems that displays the relationship between all accounting units (for example, corporate divisions, projects, reporting groups, account numbers) and determines roll-up hierarchies.

tuition lock In PeopleSoft Enterprise Campus Solutions, a feature in the Tuition Calculation process that enables you to specify a point in a term after which students are charged a minimum (or *locked*) fee amount. Students are charged the locked fee amount even if they later drop classes and take less than the normal load level for that tuition charge.

unclaimed transaction In PeopleSoft Enterprise Incentive Management, a transaction that is not claimed by a node or participant after the allocation process has completed, usually due to missing or incomplete data. Unclaimed transactions may be manually assigned to the appropriate node or participant by a compensation administrator.

universal navigation header Every PeopleSoft portal includes the universal navigation header, intended to appear at the top of every page as long as the user is signed on to the portal. In addition to providing access to the standard navigation buttons (like Home, Favorites, and signoff) the universal navigation header can also display a welcome message for each user.

update access In PeopleSoft Enterprise Campus Solutions, a type of security access that permits the user to edit and update data.

See also *inquiry access.*

user interaction object In PeopleSoft Sales Incentive Management, used to define the reporting components and reports that a participant can access in his or her context. All Sales Incentive Management user interface objects and reports are registered as user interaction objects. User interaction objects can be linked to a compensation structure node through a compensation relationship object (individually or as groups).

variable In PeopleSoft Sales Incentive Management, the intermediate results of calculations. Variables hold the calculation results and are then inputs to other calculations. Variables can be plan variables that persist beyond the run of an engine or local variables that exist only during the processing of a section.

VAT exception Abbreviation for *value-added tax exception.* A temporary or permanent exemption from paying VAT that is granted to an organization. This terms refers to both VAT exoneration and VAT suspension.

VAT exempt Abbreviation for *value-added tax exempt.* Describes goods and services that are not subject to VAT. Organizations that supply exempt goods or services are unable to recover the related input VAT. This is also referred to as exempt without recovery.

VAT exoneration Abbreviation for *value-added tax exoneration.* An organization that has been granted a permanent exemption from paying VAT due to the nature of that organization.

VAT suspension Abbreviation for *value-added tax suspension.* An organization that has been granted a temporary exemption from paying VAT.

warehouse A PeopleSoft data warehouse that consists of predefined ETL maps, data warehouse tools, and DataMart definitions.

work order In PeopleSoft Services Procurement, enables an enterprise to create resource-based and deliverable-based transactions that specify the basic terms and conditions for hiring a specific service provider. When a service provider is hired, the service provider logs time or progress against the work order.

worker A person who is part of the workforce; an employee or a contingent worker.

workset A group of people and organizations that are linked together as a set. You can use worksets to simultaneously retrieve the data for a group of people and organizations and work with the information on a single page.

worksheet A way of presenting data through a PeopleSoft Business Analysis Modeler interface that enables users to do in-depth analysis using pivoting tables, charts, notes, and history information.

worklist The automated to-do list that PeopleSoft Workflow creates. From the worklist, you can directly access the pages you need to perform the next action, and then return to the worklist for another item.

XML link The XML Linking language enables you to insert elements into XML documents to create a links between resources.

XML schema An XML definition that standardizes the representation of application messages, component interfaces, or business interlinks.

XPI Abbreviation for *eXtended Process Integrator.* PeopleSoft XPI is the integration infrastructure that enables both real-time and batch communication with EnterpriseOne applications.

yield by operation In PeopleSoft Manufacturing, the ability to plan the loss of a manufactured item on an operation-by-operation basis.

zero-rated VAT Abbreviation for *zero-rated value-added tax.* A VAT transaction with a VAT code that has a tax percent of zero. Used to track taxable VAT activity where no actual VAT amount is charged. Organizations that supply zero-rated goods and services can still recover the related input VAT. This is also referred to as exempt with recovery.

Index

A

A2AChat class
  constructor 154, 288
  fields 155, 288
  method 157
  methods 288
A2AChat class fields
  address 155
  agentID 155
  agentName 155
  appData 155
  chatType 156
  id 156
  jr 156
  type 156
  uniqueueId 157
A2AChat class method 157
A2AChat constructor 154, 288
A2AChat object 288
A2AChat.PS_JR 190, 302
A2AChat.TYPE_ANSWER 190, 302
A2AChat.TYPE_CONSULT 191, 302
A2AChatAddress class
  callback event methods 289
  constructor 157, 288
  event method 159
    See Also A2AChatAddress class callback event methods
  fields 158, 288
  method 158
  methods 289
A2AChatAddress class callback event methods
  onChatEnded 159
  onInitiatingChat 159
  onNotify 159
A2AChatAddress class fields
  id 158
  tasks 158
A2AChatAddress constructor 157, 288
A2AChatAddress object 288
acceptTask 149, 286
ACD, *See* Automatic Call Distributor (ACD)
additional documentation xxviii
addKeyValue
  AppData class 163, 290
  CallData class 168
  CallDataclass 293
  UserData class 259
address
  A2AChat class 155, 288
  Chat class 171, 294
_Address callback event methods 285
_Address class
  callback event method 147
  constructor 146
  fields 147
_Address class fields
  caps 147
  id 147
_Address constructor 147, 285
_Address fields 285
_Address object 285
_Address() constructor 285
addresses
  Session class 231, 313
  User class 243, 316
agent chat window 115
agent console, *See* MultiChannel Console
Agent Info page 32
Agent page 89
Agent State Summary component (MCF_AGENTSTATE_CMP) 102
Agent State Summary page 104
agent statistics
  monitoring 83
agentId
  A2AChat class 288
  Chat class 171, 294
  Email class 175, 296
  GenericTask class 189, 302
  _User class 287
agentID
  A2AChat class 155
  _User class 152
agentName 155, 288
agentPassword 244, 316
agents
  activating/viewing chats 114

assigning tasks 77
See Also tasks
assigning to queues 90
communicating by chat 4
communicating via chat 115
configuring CTI agents 28
configuring MCF agents 89
configuring windows 94
creating 89
customizing presence 91
defining 89
defining agent-specific URLs 96
defining optional characteristics 92
deleting 90
deleting queues 91
displaying number of agents logged into queues 83
inviting to chats 117
logging the status 83
monitoring 137
moving 100
moving physical queues 101
personalizing chat 95
selecting buddies 114
selecting state masks 24
setting maximum timeouts for queue server logout 85
setting maximum workload 91
setting presence 114
setting skill level 90
setting trace level 97
setting up buddy lists 93
specifying log/chat/buddy list names 90
specifying miscellaneous parameters 97
specifying supported languages 91
viewing the state summary 104
AgentStatistics class
constructor 160, 289
fields 160, 289
See Also AgentStatistics class fields
AgentStatistics class fields
averageCallDuration 160
averageHoldDuration 160
callsHandled 160
data 161
unavailableDuration 161
waitDuration 161
AgentStatistics constructor 160
AgentStatistics object 289
AgentStatistics() constructor 289
alternate 203, 306
ani 167, 292
answer 203, 306
AOL Instant Messenger
configuring servers 279
detecting presence 279
setting the server IP address 280
updating presence 282
appData
A2AChat class 155, 288
Chat class 171, 294
Email class 175, 296
GenericTask class 189, 302
AppData class 162
constructor 161, 290
fields 161, 290
See Also AppData class fields
methods 163, 290
AppData class fields
data 162
groupId 162
jr 162
question 162
strData 162
subject 162
uniqueId 162
url 163
userId 163
username 163
wizUrl 163
AppData constructor 161, 290
AppData object 290
application fundamentals xxvii
application servers
chat architecture 6
See Also chat
logging 110
MCF cluster architecture 65
MCF servers 69
See Also MCF servers
notifying MCF clusters of shutdown 87
queue servers 71
See Also queue servers
REN servers 49
See Also REN servers
specifying the authentication domain 57
Aspect 21, 41

attachUserData 204, 306
authentication domain 61
  configuring REN servers 56, 57
Automatic Call Distributor (ACD)
  configuring CTI queues 28
  CTI architecture 12
    See Also CTI
  switching agent ready status 45
  viewing the ACD group in the CTI console 40
availableCost 260, 318
Avaya Definity G3 21
averageCallDuration 160, 289
averageHoldDuration 160, 289
averageTaskDuration 200, 305
averageWaitTime 200, 305

B

backup queue servers 71
Balance Queue page 102
browsers
  authentication domain 56
  browsers required for CTI 16
  configuring REN server security 51
  MCF browser windows 6
  MCF cluster browser URLs 68, 71
  MultiChannel Console 6
  REN server cluster requests 51
  setting REN server browser URLs 60
buddies 232, 313
Buddy class
  callback event methods 291
  constructor 164, 291
  event method 165
    See Also Buddy class callback event methods
  fields 291
Buddy class callback event methods
  onStat1 165
  onStat2 165
  onState 165
Buddy constructor 164, 291
Buddy List page 93
buddy lists
  selecting buddies in the MultiChannel Console 114
  setting up for agents 93
Buddy object 291
buffer tests, REN server 60, 71
built-in functions, PeopleCode 142
Button Sample page 282

C

call 202, 306
  Line class 306
Call class
  constructor 165
  fields 166, 292
    See Also Call class fields
Call class fields 166
  line 166
  statistics 167
Call constructor 166, 291
Call object 291
callback event
  onNotify
    _UQAddress class 286
  onTaskRemoved 286
callback event methods
  A2AChatAddress class 159, 289
  _Address class 147, 285
  Buddy class 165, 291
  ChatAddress class 174, 295
  EmailAddress class 297
  Extension class 182, 298
  GenericAddress class 300
  Group class 195, 303
  Line class 212, 307
  onAccepted 286
  onAcceptingTask 286
  onAddressRegistered 314
  onAddressUnregistered 314
  onAlternating 307
  onAnswering 307
  onAttachingUD 307
  onBroadcast 312
  onBuddyRegistered 314
  onBuddyUnregistered 314
  onCallDataChanged 307
  onCancelingDnd 298
  onCancelingForward 298
  onCapabilitiesChanged 307, 316
  onChatEnded 289
  onClearing 307
  onClosed 314
  onCompleting 307
  onConferencing 307
  onCtiBusy 316
  onCtiClear 316
  onDequeueingTask 286

onDialing 307
onDnd 298
onDndCanceled 298
onDown 311
onDropped 307, 317
onError 307, 314
_Address class 285
User class 317
onForwardCanceled 298
onForwarded 298
onForwarding 298
onGrabbing 307
onGroupRegistered 314
onGroupUnregistered 314
onHbLost 312
onHbRecovered 312
onHeld 307
onHolding 307
onHook 308
onInfo 314, 317
onInitiatingChat 289
onInService 312
onLoggedIn 317
onLoggedOut 317
onLoggingIn 317
onLoggingOut 317
onMediaDisabled 317
onMediaEnabled 317
onNotify
A2AChatAddress class 289
onNotReady 317
onOffHook 308
onOpened 314
onOutOfService 312
onParking 308
onPartyAdded 308
onPartyChanged 308
onPartyRemoved 308
onPresenceChanged 317
onReady 317
onReconnecting 308
onRegistered 317
onRegistering 317
onReleasing 308
onRestart 312
onRetrieving 308
onRinging 308
onSettingDnd 298
onSettingNotReady 317
onSettingPresence 317
onSettingReady 317
onSettingWorkNotReady 317
onSettingWorkReady 317
onStat
Group class 303
User class 317
onStat1
Buddy class 291
Group class 303
onStat1(
User class 317
onStat2
Buddy class 291
Group class 303
User class 317
onState 291
onTalking 308
onTaskAdded 303
_UQAddress class 286
onTaskRemoved 303
onTransfering 308
onUnassigned
_UQAddress class 286
onUnknown 317
onUnregistered 317
onUnregistering 317
onUp 311
onUpdatingCD 308
onUserDataChanged 308
onUserRegistered 314
onUserUnregistered 314
onWorkNotReady 317
onWorkReady 317
RenServer class 228, 311
Server class 230, 312
Session class 237, 314
_UQAddress class 149, 286
User class 251, 316
CallData class
constructor 167, 292
fields 167, 292
method 168
methods 293
CallData class fields
ani 167
callId 167
callType 168
data 168
dnis 168
CallData constructor 167, 292

CallData object 292
callId 167, 292
calls
  answering 42
  completing 46
  CTI architecture 11
    See Also CTI
  dialing outbound 45
  disconnecting callers 45
  holding/retrieving 44
  initiating conference calls 44
  selecting actions in the CTI console 40
  selecting agent status for incoming 45
  selecting masks 25
  testing outbound 36
  transferring callers 42
  understanding call actions 41
callsHandled 160, 289
CallStatistics class
  constructor 169, 293
  fields 169, 293
CallStatistics class fields
  data 169
  holdTime 169
  queueTime 169
  talkTime 170
CallStatistics constructor 169, 293
CallStatistics object 293
callType 168, 292
canAlternate 218, 308
canAnswer 218, 308
canAttachUserData 308
canAttachUserDate 218
canCancelDnd 184, 299
canCancelDndFwdDefault 299
cancelDnd 179, 298
cancelForwardSet 179, 298
canClear 218, 308
canComplete 218, 308
canConference 218, 309
canConferenceSingle 219, 309
canDial 184, 299
canDropParty 219, 309
canFwdBusy 184, 299
canFwdBusyNoAnswer 184, 299
canFwdCancelForward 184, 299
canFwdDefault 184
canFwdNoAnswer 185, 299
canFwdUnconditional 185, 299
canHold 219, 309
canLogin 257
canLogout 257
canPark 219, 309
canReconnect 219, 309
canRefreshState 185, 257, 299
canRelease 219, 309
canRetrieve 219, 309
canSetDnd 185, 299
canSetNotReady 257
canSetPresence 258
canSetReady 258
canSetWorkNotReady 258
canSetWorkReady 258
canTransfer 220, 309
canTransferMute 220, 309
canUpdateCallData 220, 309
caps
  _Address class 147, 285
  Line class 202
  _User class 152, 287
caseid 240, 314
channels
  communicating by chat 4
    See Also chat
  email 265
    See Also email channels
  generic 5
  instant messaging 5
    See Also instant messaging
  MultiChannel Console 6
  understanding 3
  voice 4
chat
  activating 114
  architecture 6
  collaborating by 4
  communicating with agents 115
  communicating with customers 117
  defining responses 75
  ending 117, 118
  forwarding to other queues 117
  initiating 114
  instant messaging 279
    See Also instant messaging
  inviting agents 117
  logging 68, 85
  moving agents during ongoing sessions 101
  personalizing 95
  responding 117, 118

selecting task acceptance mode 94
sending template messages 117
sending/grabbing URLs 117
setting up buddy lists 93
specifying agent names 90
specifying greeting interval 80
using in MultiChannel Console 113
using the Customer Chat Sample page 122
viewing history 116
viewing logs 106
Chat class
constructor 170, 293
fields 170, 293
See Also Chat class fields
methods 172, 294
Chat class fields 170
address 171
agentId 171
appData 171
customerName 171
groupId 171
question 172
subject 172
Chat constructor 170, 293
Chat Log component (MCF_CHAT_LOG_CMP) 105
Chat object 293
Chat page 106
Chat Responses page 75
ChatAddress class
callback event methods 295
constructor 172, 294
event method 174
fields 294
methods 295
ChatAddress constructor 173, 294
ChatAddress object 294
chatType 156, 288
Cisco
components required for CTI 16
configuring the expanded call context variable 26
CTI architecture 11
See Also CTI
implementing 24
logging out 32
selecting a CTI vendor 21
selecting a switch 21
setting agent state masks 24
setting CTI client signature 29
setting masks 24
setting the heartbeat 24
setting up for pop-ups 34
switch limitations 38
transferring callers 43
clear 204, 306
close 233, 313
closeSession 224, 310
Cluster Notify component (MCF_AD_NOTIFY_CMP) 86
Cluster Summary component (MCF_RSERV_CFG_CMP) 81
Cluster Summary page 81
Cluster Tuning component (MCF_SYSTEM_NV_CMP) 81
Cluster Tuning page 82
clusters
MCF 65
See Also MCF clusters
REN server 69
See Also REN servers
code 226, 311
comments, submitting xxxii
common elements xxxiii
complete 205, 306
computer telephony integration (CTI), *See* CTI
conference 205, 306
conferenceSingle 205, 306
Configuration page (instant messaging) 280
connectionid 202
connectionId 306
Connectors page 267
consoles
agent 4
See Also MultiChannel Console
CTI 11
See Also CTI console
Java 29
MultiChannel 113
See Also MultiChannel Console
constructor
A2AChat class 288
A2AChat Class 154
A2AChatAddress class 157, 288
_Address class 146, 285
AgentStatistics 160
AgentStatistics class 289

AppData class 161, 290
Buddy class 164, 291
Call class 165, 291
CallData class 167, 292
CallStatistics class 293
CallStatisticsclass 169
Chat class 170, 293
ChatAddress 172
ChatAddress class 294
Email class 174, 295, 296
EmailAddress class 176
Extension class 177, 297
ExtensionCaps class 183, 299
ForwardMode class 185, 299
GenericAddress class 186, 300
GenericTask class 188, 301
Group class 193, 303
GroupStatistics class 196, 304
GroupStatistics1 class 197, 304
GroupStatistics2 class 200, 305
Line class 201, 305
LineCaps class 217, 308
MCEvent class 220, 309
MediaType class 221, 310
PSMC class 222, 310
Reason class 226, 311
RenServer class 227, 311
Server class 228, 312
Session class 231, 312
Task class 240, 314
_UQAddress class 148, 286
User class 242, 315
_User class 151, 287
UserCaps class 256, 257
See Also UserCaps class fields
UserData class 258, 318
UserStatistics1 class 260, 318
UserStatistics2 class 262, 319
contact information xxxii
cost 240, 314
cross-references xxxi
CTI
accessing 39
agents 18, 27
See Also CTI agents
answering calls 42
applying applet protection 16
Cisco components, required 16
Cisco switch limitations 38
completing calls 46
components 12
configuring 16
configuring pop-up windows 32
configuring queues 28
console 40
See Also CTI console
creating configurations 19
dialing outbound calls 45
disconnecting callers 45
enabling 18
event logging 35
Genesys components, required 15
holding calls 44
implementing Cisco 24
See Also Cisco
implementing free seating 35
implementing Genesys 23
See Also Genesys
initiating conference calls 44
installing 16, 18
JRE, required 16
log 85
miscellaneous 16
monitoring 83
outbound call sample page 35
personalizing 31
phone books 30
phone books, shared 16, 22
queues 27
registering with middleware 40
sample page 35
setting pop-up screens 26
supporting single signon 35
testing calls 36
transferring callers 42
understanding 11, 37
understanding call actions 41
understanding components 11
using 38
using hot keys 46
using the interface 38
CTI Agent Configuration component (PT_CTI_AGENT) 27
CTI Agent Configuration page 28
CTI agents
configuring 28
logging out 32, 41
phone books 30
phone books, shared 22
signing on to PeopleSoft 18

switching agent ready status 45
viewing information 32
viewing status 40
wrap-up mode 45
CTI Applet page 19
CTI Cisco page 24
CTI configuration component (PT_CTI_CONFIG) 16
CTI console 12
applet 16, 20
browser requirements 16
components 11
floating 32
heartbeat interval 83
JSMCAPI 16
non-applet 16, 19
synchronizing with teleset 43
understanding 11
using 40
CTI Genesys page 23
CTI Non-Applet page 19
CTI queues, configuring 28
CTI Sample Pages component (PT_CTI_DEMOOUTB) 129
CTI Sample Pages component (PT_CTI_DEMOUTB) 35
CTI Type page 19
ctiBusy 260, 318
currentQueue 261, 263, 319
Customer Chat page 122
customer chat window 118
Customer Connection website xxviii
customerid 241, 314
customerName
Chat class 171, 294
Email class 296
GenericTask class 189, 302
customers
accepting tasks in chat 94
chat architecture 6
See Also chat
communicating by chat 4
communicating via chat 117
using CTI 37
See Also CTI
using the Customer Chat Sample page 122
Customize Presence page 91

D

data
AgentStatistics class 161, 289
AppData class 162, 290
CallData class 168, 292
CallStatistics class 169, 293
GroupStatistics class 196, 304
debugging
logging for application servers 110
REN servers 56
tracing level for agents 97
troubleshooting CTI 29
viewing event logs 107
defining
MCF queues 74
Definity G3, Avaya 21
DeQueue
demonstrating 127, 129
understanding 142
dequeueTask 149, 286
desc 227
description 311
dial 206, 306
digital signatures 16
disableMedia 244, 316
dnis 168, 292
documentation
printed xxviii
related xxviii
updates xxviii
domain
authentication domain 61
domains
application server domains 65
See Also application servers
configuring MCF server 69
See Also MCF
MCF cluster architecture 65
setting screen pop-up URLs 26
setting the authentication token domain for REN servers 56
See Also authentication domain
done list, size 83
dropParty 206, 306

E

ECC variables 34
email

accessing/deleting from the database 276
authorizing attachments for viewing/deleting 276
channels 265
See Also email channels
defining connector and mail server properties 268
email IDs 277
GetMail - Server page 274
handling headers, size determination and time zone offsets 266
HTML 5
instant messaging 279
See Also instant messaging
MailStore - DB page 276
MIME 273
See Also Multipurpose Internet Mail Extensions (MIME)
processing architecture 8
reading/deleting from mail servers 274
setting access, attachment, language and threshold properties 268
storing in/retrieving from database servers 274
understanding 5
unique identifiers (UIDs) 276
universal queue 5
using sample pages 127
using the Email window 128
email channels
configuring GETMAILTARGET 267
configuring Integration Broker 266
demonstrating 273
understanding 265
Email class
constructor 174, 295, 296
fields 175, 295
See Also Email class fields
methods 175, 296
Email class fields 175
agentId 175
appData 175
Email constructor 174, 295
Email object 295
Email page 127
Email Sample Pages component (MCFEM_DEMOERMS_CMP) 273
Email window 128
EmailAddress class
callback event methods 297
constructor 176
fields 296
methods 297
EmailAddress constructor 176, 296
EmailAddress object 296
enableMedia 245, 316
enableMedialogout 246
enableMediasetWorkNotReady 249
EnQueue
accepting tasks in chat 94
demonstrating generic persistent events 125
setting agent languages 91
understanding 142
errors
determining email size 266
logging for application servers 110
REN servers 56
troubleshooting CTI 29
escalation
administering tasks 112
specifying the default timeout for tasks 80
Escalation Admin (administration) page 112
Escalation Administration component (MCF_ESCAL_CMP) 111
event, *See* callback event methods
A2AChatAddress class callback 159
_Address class callback 147
Buddy class callback 165
ChatAddress class callback 174
Extension class callback 182
Group class callback 195
Line class callback 212
RenServer class callback 228
Server class callback 230
Session class callback 237
_UQAddress class callback 149
User class callback 251
Event Log component (MCF_EVENTLOGL_CMP) 105
Event Log page 108
events
CTI event logging 35
Generic Event page 125
Generic Event window 126
MCF log server failover 68
real-time event notification 49

See Also REN servers
types 108
viewing event logs 107
Expanded Call Context (ECC)
variables 34
extension 221, 309
Extension class
callback event methods 298
constructor 177, 297
event method 182
See Also Extension class callback event methods
fields 178, 297
See Also Extension class fields
method 179
methods 298
Extension class callback event methods
onCancelingDnd 182
onCancelingForward 182
onDnd 182
onForwardCanceled 182
onForwarded 183
onForwarding 183
onSettingDnd 183
Extension class fields 178
forwardMode 178
isDnd 178
lines 178
numOfLines 179
Extension class method
cancelDnd 179
cancelForwardSet 179
forwardSet 180
getDialingLine 180
getFreeLine 180
getLineByConnectionId 181
getOffHookLine 181
setDnd 181
Extension constructor 177, 297
Extension object 297
ExtensionCaps class
constructor 299
fields 184, 299
See Also ExtensionCaps class fields
ExtensionCaps class fields
canCancelDnd 184
canDial 184
canFwdBusy 184
canFwdBusyNoAnswer 184
canFwdCancelForward 184
canFwdDefault 184
canFwdNoAnswer 185
canFwdUnconditional 185
canRefreshState 185
canSetDnd 185
Extensioncaps constructor 299
ExtensionCaps constructor 183
ExtensionCaps object 299

F

failover
configuring REN servers 49
MCF log servers 68
queue servers 67
REN server clusters 51
understanding 65
fields
A2AChat class 288
A2AChat Class 155
See Also A2AChat class fields
A2AChat.PS_JR 302
A2AChat.TYPE_ANSWER 302
A2AChat.TYPE_CONSULT 302
A2AChatAddress class 158, 288
See Also A2AChatAddress class fields
address
A2AChat class 288
Chat class 294
_Address class 147, 285
See Also _Address class fields
addresses
Session class 313
User class 316
agentId
A2AChat class 288
Chat class 294
Email class 296
GenericTask class 302
_User class 287
agentName 288
agentPassword 316
AgentStatistics class 160, 289
ani 292
appData
A2AChat class 288
Chat class 294
Email class 296
GenericTask class 302
AppData class 161, 290
availableCost 318

averageCallDuration 289
averageHoldDuration 289
averageTaskDuration 305
averageWaitTime 305
buddies 313
Buddy class 291
call 306
  Line class 306
Call class 166, 292
CallData class 167, 292
  See Also CallData class fields
callId 292
callsHandled 289
CallStatistics class 169, 293
  See Also CallStatistics class fields
callType 292
canAlternate 308
canAnswer 308
canAttachUserData 308
canCancelDnd 299
canClear 308
canComplete 308
canConference 309
canConferenceSingle 309
canDial 299
canDropParty 309
canFwdBusy 299
canFwdBusyNoAnswer 299
canFwdCancelForward 299
canFwdDefault 299
canFwdNoAnswer 299
canFwdUnconditional 299
canHold 309
canPark 309
canReconnect 309
canRefreshState 299
canRelease 309
canRetrieve 309
canSetDnd 299
canTransfer 309
canTransferMute 309
canUpdateCallData 309
caps
  _Address class 285
  _User class 287
caseid 314
Chat class 170, 293
ChatAddress class 294
chatType 288
code 311
connectionId 306
cost 314
ctiBusy 318
currentQueue 319
customerid 314
customerName
  Chat class 294
  Email class 296
  GenericTask class 302
data
  AgentStatistics class 289
  AppData class 290
  CallData class 292
  CallStatistics class 293
  GroupStatistics class 304
description 311
dnis 292
Email class 175, 295
EmailAddress class 296
extension 309
Extension class 178, 297
ExtensionCaps class 184, 299
forwardMode 298
ForwardMode class 186, 300
GenericAddress class 187, 188, 300
GenericTask class 301
getUrl
  GenericTask class 302
GLOBALS class 190, 302
group 309, 314
Group class 193, 303
groupId 290, 294
  GenericTask class 302
groups 313
GroupStatistics class 196, 304
GroupStatistics1 class 197, 304
GroupStatistics2 class 200, 305
holdTime 293
id
  A2AChat class 288
  A2AChatAddress class 289
  _Address class 285
  Group class 303
  Line class 306
  Server class 312
  Session class 313
  Task class 314
  _User class 287
info
  Server class 312

isDnd 298
isRunning 311
jr
A2AChat class 288
AppData class 290
language 316
line 292
Line class 202, 306
LineCaps class 218, 308
lines 298
MCEvent class 220, 309
MediaType class 222, 310
mode 300
mostRecentTaskData 319
mostRecentTaskDone 304, 305
mostRecentTaskDoneData 304
mostRecentTaskEnqueued 304
mostRecentTaskEnqueuedData 304
mostRecentTaskId 319
name
Group class 303
_User class 287
numAgentsAvailable 304
numAgentsLoggedIn 304
numberOfAbandoned 304
numberOfLoggedIn 304
numberOfQueued 304
numEscalation 304
numOfLines 298
numOverflow 304
numTaskAccepted 304
UserStatistics1 class 319
numTaskDone 305
numTasksDone 319
numTasksUnassigned 319
oldestTask 305
onStat 314
presences 287
presenceText 319
priority 315
PSMC class 223, 310
question
AppData class 290
Chat class 294
GenericTask class 302
queuedWaitTime 304
queueTime 293
reason 309
Reason class 226, 311
reasonFlag
GroupStatistics1 class 305
UserStatistics1 class 319
recentTask 305
registeredAddresses 316
relativeQueueLoad 304
renServer 310
RenServer class 227, 311
Server class 229, 312
Server.TYPE.CTI 302
Server.TYPE_UQ 302
serverID 313
servers 310
Session class 231, 313
sessions 310
ST_LOGGEDIN 287
ST_LOGGEDOUT 287
ST_NOTREADY 287
ST_READY 287
ST_UNKNOWN 287
ST_WORKNOTREADY 287
ST_WORKREADY 287
state
Line class 306
Server class 312
Session class 313
UserStatistics1 class 319
states 287
statistics
Call class 292
Group class 303
_User class 287
statistics1
Group class 303
_User class 287
statistics2
Group class 303
_User class 287
strData 290
subject
AppData class 290
Chat class 294
GenericTask 302
talkTime 293
Task class 240, 314
Task.TYPE_A2ACHAT 302
Task.TYPE_CHAT 302
Task.TYPE_CTI 302
Task.TYPE_EMAIL 302
Task.TYPE_GENERIC 302
tasks

A2AChatAddress class 289
_UQAddress class 286
taskTotalTimeInSystem 305
timeElapsedOldestTask 305
timeElapsedRecentTask 305
timeIdle 319
timeInCurrentState 319
timeNotReady 319
timeSinceLoggedIn 319
timeSinceLogin 319
timeSinceStart 305
totalTimeAvailable 319
totalTimeUnavailable 319
type
A2AChat class 288
MediaType class 310
Server class 312
Task class 315
unavailableDuration 290
uniqueId 290
uniqueueId 288
_UQAddress class 148, 286
url 290, 311
urlAbs 315
urlRel 315
user 309
Session class 313
User class 243, 315
_User class 152, 287
See Also _User class fields
userData 292
UserData class 259, 260
See Also UserStatistics1 class fields
userId 290
username 290
UserStatistics1 class 318
UserStatistics2 class 263, 319
See Also UserStatistics2 class fields
waitDuration 290
wizUrl 290
floating CTI console 31, 32
Forward function
demonstrating 128
understanding 142
forwardMode 178, 298
ForwardMode class
constructor 185, 299
fields 186, 300
ForwardMode constructor 185, 299
ForwardMode object 299
forwardSet 180, 298
free seating 35

G

Gateways page 266
gateways, configuring for email channels 266
Generic Event page 125
Generic Event window 126
GenericAddress class
callback event methods 300
constructor 186, 300
fields 187, 188, 300
See Also GenericTask class Fields
method 187
methods 300
GenericAddress constructor 186, 300
GenericAddress object 300
GenericTask class
constructor 188, 301
fields 301
method 190
methods 302
GenericTask class fields 188
agentId 189
appData 189
customerName 189
groupId 189
question 189
subject 189
GenericTask constructor 188, 301
GenericTask object 301
Genesys
components required for CTI 15
CTI architecture 11
See Also CTI
implementing 23
logging out 41
selecting a CTI vendor 21
selecting a switch 21
setting up for pop-ups 33
wrap-up time 46
getAni 207, 306
getCallById 224, 310
getDescr 207, 306
getDialingLine 180, 298
getDnis 207, 306
getFreeLine 180, 298
getLineByConnectionId 181, 298
getLineById 224, 310

GetMail - Server page 274
GETMAILTARGET, configuring 267
getOffHookLine 181, 298
getReferenceId 208, 306
getUrl
A2AChat class 288
Chat class 172, 294
Email class 175, 296
GenericTask class 190, 302
Line class 208, 307
getURL
A2AChat class 157
GLOBALS class
fields 190, 302
See Also GLOBALS class fields
method 192
methods 303
GLOBALS class fields
A2AChat.PS_JR 190
A2AChat.TYPE_ANSWER 190
A2AChat.TYPE_CONSULT 191
Server.TYPE_CTI 191
Server.TYPE_UQ 191
Task.TYPE_A2ACHAT 191
Task.TYPE_CHAT 191
Task.TYPE_CTI 192
Task.TYPE_EMAIL 192
Task.TYPE_GENERIC 192
GLOBALS class methods 192
GLOBALS class methods 192
isValid 193
glossary 321
grabCall 208, 307
group 309, 314
MCEvent class 221
Task class 241
Group class 194
callback event methods 303
constructor 193, 303
event method 195
See Also Group class callback event methods
fields 193, 303
See Also Group class fields
Group class callback event methods
onStat 195
onStat1 195
onStat2 195
onTaskAdded 195
onTaskRemoved 195
Group class fields
id 194
name 194
registered 194
statistics 194
statistics1 194
statistics2 194
Group constructor 193, 303
groupId 290, 294
AppData class 162
Chat class 171
GenericTask class 189, 302
groups 232, 313
GroupStatistics class
constructor 196, 304
fields 196, 304
See Also GroupStatistics class fields
GroupStatistics class fields
data 196
numberOfAbandoned 196
numberOfLoggedIn 196
numberOfQueued 197
queuedWaitTime 197
relativeQueueLoad 197
GroupStatistics constructor 196, 304
GroupStatistics object 304
GroupStatistics1 class 305
constructor 197, 304
fields 197, 304
See Also GroupStatistics1 class fields
GroupStatistics1 class fields
mostRecentTaskDone 198
mostRecentTaskDoneData 198
mostRecentTaskEnqueued 198
mostRecentTaskEnqueuedData 198
numAgentsAvailable 198
numAgentsLoggedIn 198
numEscalation 198
numOverflow 199
numTaskAccepted 199
numTaskDone 199
numTaskQueued 199
reasonFlag 199
taskTotalTimeInSystem 199
timeSinceStart 199
GroupStatistics1 constructor 197, 304
GroupStatistics1 object 304
GroupStatistics2 class
constructor 200, 305
fields 200, 305

See Also GroupStatistics2 class fields
GroupStatistics2 class fields
averageTaskDuration 200
averageWaitTime 200
oldestTask 200
recentTask 201
timeElapsedOldestTask 201
timeElapsedRecentTask 201
GroupStatistics2 constructor 200, 305
GroupStatistics2 object 305

H

heartbeat interval, JSMCAPI client 83
heartbeat, setting Cisco 24
Hicom 300, Siemens 21
hold 209, 307
hold status 44
holdTime 169, 293
hot keys
using CTI hot keys 46
using MultiChannel Console 115

I

id 152, 194
A2AChat class 156, 288
A2AChatAddress class 158, 289
_Address class 147, 285
Group class 303
Line class 202, 306
Server class 229, 312
Session class 232, 313
Task class 241, 314
_User class 287
IMAP4, *See* Internet Message Access Protocol 4 (IMAP4)
implementation
MultiChannel Framework 1
info 229
Server class 312
InitChat
demonstrating 122
setting agent languages 91
understanding 143
initiateChat 158, 289
initJSMCAPI 303
instant messaging
AOL Instant Messenger 279, 280
See Also AOL Instant Messenger
configuring 280
configuring servers 279
demonstrating 281, 282
integrating 279
Lotus Sametime Connect 279
See Also Lotus Sametime Connect
understanding 5, 279
See Also chat
using sample pages 281
Yahoo! Messenger 279, 280
See Also Yahoo! Messenger
Instant Messaging Configuration component (MCF_IM_CFG_CMP) 280
Instant Messaging Sample Pages component (MCF_IM_DEMO_CMP) 281
Integration Broker
configuring for email channels 266
setting the email size threshold 271
Interactive Voice Response (IVR)
CTI architecture 12
See Also CTI
Internet Message Access Protocol 4 (IMAP4)
calculating email size 273
setting the connector threshold 270
isDnd 178, 298
isMediaEnabled 245, 316
isRunning 227, 311
isValid 193, 303
IVR, *See* Interactive Voice Response (IVR)

J

Java Console 29
Java Runtime Environment (JRE), *See* JRE
JavaScript MultiChannel API (JSMCAPI), *See* JSMCAPI
jr 162
A2AChat class 156, 288
AppData class 290
JRE 16
JSMCAPI 12, 15
JSMCAPI classes
A2AChat class 287
A2AChatAddress class 288
_Address class 285
AgentStatistics class 289
AppData class 290
Buddy class 290
Call class 291
CallData class 292
CallStatistics class 293

Chat class 293
ChatAddress class 294
Email class 295
EmailAddress class 296
Extension class 297
ExtensionCaps class 299
ForwardMode class 299
GenericAddress class 300
GenericTask class 301
GLOBALS class 302
Group class 303
GroupStatistics class 303
GroupStatistics1 class 304
GroupStatistics2 class 305
Line class 305
LineCaps class 308
MCEvent class 309
MediaType class 309
PSMC class 310
Reason class 311
RenServer class 311
Server class 311
Session class 312
Task class 314
_UQAddress class 285
User class 315
_User class 286
UserData class 318
UserStatistics1 class 318
UserStatistics2 class 319

L

language 244, 316
languages
  setting agent support for 91
  setting for email 271
Languages page 91
line 166, 292
Line class
  callback event methods 307
  constructor 201, 305
  event method 212
    See Also Line class callback event methods
  fields 202, 306
    See Also Line class fields
  method 203
  methods 306
Line class callback event methods
  onAlternating 212
  onAnswering 212
  onAttachingUD 212
  onCallDataChanged 213
  onCapabilitiesChanged 213
  onClearing 213
  onCompleting 213
  onConferencing 213
  onDialing 213
  onDropped 214
  onError 214
  onGrabbing 214
  onHeld 214
  onHolding 214
  onOffHook 215
  onOnHook 215
  onParking 215
  onPartyAdded 215
  onPartyChanged 215
  onPartyRemoved 215
  onReconnecting 216
  onReleasing 216
  onRetrieving 216
  onRinging 216
  onTalking 216
  onTransfering 217
  onUpdatingCD 217
  onUserDataChanged 217
Line class fields
  call 202
  caps 202
  connectionid 202
  id 202
  state 202
Line class methods
  alternate 203
  answer 203
  attachUserData 204
  clear 204
  complete 205
  conference 205
  conferenceSingle 205
  dial 206
  dropParty 206
  getAni 207
  getDescr 207
  getDnis 207
  getReferenceId 208
  getUrl 208
  grabCall 208
  hold 209

park 209
reconnect 210
release 210
retrieve 210
transfer 211
transferMute 211
updateCallData 212
Line constructor 201, 305
Line object 305
LineCaps class
constructor 217, 308
fields 218, 308
See Also LineCaps class fields
LineCaps class fields
canAlternate 218
canAnswer 218
canAttachUserDate 218
canClear 218
canComplete 218
canConference 218
canConferenceSingle 219
canDropParty 219
canHold 219
canPark 219
canReconnect 219
canRelease 219
canRetrieve 219
canTransfer 220
canTransferMute 220
canUpdateCallData 220
LineCaps constructor 217, 308
LineCaps object 308
lines 178, 298
log servers, MCF, *See* MCF log servers
LogFence settings 110
logging
agent status 83
application servers 110
chat 85, 116
creating log servers 70
CTI 85
CTI events 35
MCF log server failover 68
queue server status 84
REN servers 56, 84
specifying agent names for logs 90
statistics 84
tracing level for agents 97
viewing chat logs 106
viewing event logs 107
logical queues
assigning agents 90
balancing agents/tasks across 102
deleting 74
moving agents 100
moving physical queues within 101
scalability 69
understanding 67
See Also queues
viewing physical queue IDs 75
login 245, 316
loginUq 246, 316
logout 316
logoutUq 247, 316
Lotus Sametime Connect
configuring instant messaging servers 279
detecting presence 282
instant messaging 279
setting the server IP address 280

M

mail servers
configuring GETMAILTARGET properties 267
reading/deleting email 274
MailStore - DB page 276
masks, Cisco 24
master and slave servers
determining master MCF server failure 85
MCF log server failover 68
queue server failover 67
understanding MCF cluster servers 66
master servers, *See* master and slave servers
MCEvent class
constructor 220, 309
fields 220, 309
See Also MCEvent class fields
MCEvent class fields
extension 221
group 221
reason 221
user 221
MCEvent constructor 220, 309
MCEvent object 309
MCF, *See* MultiChannel Framework (MCF)
agent monitoring 137
sample pages 121

MCF Agent component (MCF_AGENT_ CMP) 89
MCF agents, *See* agents
MCF Cluster component (MCF_UQCLUSTER) 70
MCF Cluster page 70
MCF clusters
  architecture 65
  configuring 65, 68, 69, 70
  defining 70
  deleting 71
  notifying of changed parameters 86
  queue server scalability 69
  queue servers within 71
  tuning parameters 81
  understanding 66
  viewing cluster IDs 75
  viewing summary information 81
MCF log servers
  configuring MCF clusters 68
  creating 70
  failover 68
  understanding MCF clusters 66
MCF Queue component (MCF_Q_ CONFIG_CMP) 73
MCF queue servers
  creating 70
MCF queues
  configuring and defining 73
  defining 74
  defining chat responses 75
  defining static push URLs 76
MCF Sample Pages component (MCF_DEMO_CMP) 121, 129
MCF servers
  architecture 6
  configuring 69
  determining master failure 85
MCF Task component (MCF_TASKCFG_ CMP) 77
MCF_ACCPT_TASK_CMP component 99
MCF_AD_NOTIFY_CMP component 86
MCF_AGENT_CMP component 89
MCF_AGENTSTATE_CMP component 102
MCF_CHAT_LOG_CMP component 105
MCF_DEMO_CMP component 121, 129
MCF_ESCAL_CMP component 111
MCF_EVENTLOGL_CMP component 105
MCF_IM_CFG_CMP component 280
MCF_IM_DEMO_CMP component 281
MCF_OVERFLOWL_CMP component 111
MCF_Q_CONFIG_CMP component 73
MCF_QSERVSTATE_CMP component 102
MCF_QUEUESTATE_CMP 102
MCF_RSERV_CFG_CMP component 81
MCF_SYSTEM_NV_CMP component 81
MCF_TASKCFG_CMP component 77
MCF_UQCLUSTER component 70
MCFEM_DEMOERMS_CMP component 273
MediaType class
  constructor 221, 310
  fields 222, 310
MediaType constructor 222, 310
MediaType object 310
Meridian, Nortel 21
messaging, instant, *See* instant messaging
method
  _Address class callback event 147
  alternate 306
  answer 306
  attachUserData 306
  clear 306
  close 313
  closeSession 310
  complete 306
  conference 306
  conferenceSingle 306
  dial 306
  disableMedia 316
  dropParty 306
  enableMedia 316
  getAni 306
  getCallById 310
  getDescr 306
  getDnis 306
  getLineById 310
  getReferenceId 306
  getUrl
    Line class 307
  grabCall 307
  hold 307

isMediaEnabled 316
login 316
loginUq 316
logout 316
logoutUq 316
open 313
openSession 310
park 307
reconnect 307
register 316
registerAddress 313
registerBuddy 313
registerGroup 313
registerUser 313
release 307
retrieve 307
setNotReady 316
setPresence 316
setPresenceUq 316
setReady 316
setWorkNotReady 316
setWorkReady 316
start 310
statPublish 313
stop 310
transfer 307
transferMute 307
unRegister 316
UserData class 318
unregisterAddress 313
unregisterBuddy 313
unregisterGroup 313
unregisterUser 313
updateCallData 307
methods
A2AChat class 288
A2AChat Class 157
A2AChatAddress class 158, 289
A2AChatAddress class callback event 159
acceptTask 286
addKeyValue
AppData class 290
CallData class 293
AppData class 163, 290
Buddy class callback event 165
CallData class 168, 293
cancelDnd 298
cancelDndforwardSet 298
cancelForwardSet 298
Chat class 172, 294
ChatAddress class 295
ChatAddress class callback event 174
dequeueTask 286
Email class 175, 296
EmailAddress class 297
Extension class 298
Extension Class 179
See Also Extension class method
Extension class callback event 182
GenericAddress class 187, 300
GenericTask class 190, 302
getDialingLine 298
getFreeLine 298
getLineByConnectionId 298
getOffHookLine 298
getUrl
A2AChat class 288
Chat class 294
Email class 296
GLOBALS class 192, 303
See Also GLOBALS class methods
Group class callback event 195
initiateChat 289
initJSMCAPI 303
isValid 303
Line class 203, 306
See Also Line class methods
Line class callback event 212
PSMC class 223, 310
See Also PSMC class methods
RenServer class callback event 228
Server class callback event 230
Session class 233, 313
See Also Session class methods
Session class callback event 237
setDnd 298
_UQAddress 148
_UQAddress class 286
_UQAddress class callback event 149
User class 244, 316
See Also User class methods
User class callback event 251
UserData class 259, 318
MIME*, See* Multipurpose Internet Mail Extensions (MIME)
Miscellaneous component (PT_CTI_MISC) 16
Miscellaneous page 26, 97
MMA Partners xxviii

mode 300
mode field 186
Monitor Agents page 137
Monitor Queues page 139
mostRecentTaskData 261, 319
mostRecentTaskDone 198, 304, 305
mostRecentTaskDoneData 198, 304
mostRecentTaskEnqueued 198, 304
mostRecentTaskEnqueuedData 198, 304
mostRecentTaskId 261, 319
Move Agent page 100
Move Queue page 101
MultiChannel Console
  channels 4
  CTI interface 38
  displaying logged agents/unassigned tasks per queue 83
  managing tasks/chat 113
  understanding 6
  using hot keys 115
MultiChannel Framework (MCF)
  administering 99
  configuring 69
  developing for 121
  elements 3
  failover 65
  getting started 1
  implementation 1
  overview 1
  scalability 65
  server architecture 6
  understanding 3
Multipurpose Internet Mail Extensions (MIME) 5, 273

N

name
  Group class 194, 303
  _User class 287
  _User class fields 152
nodes
  configuring for email channels 266
  configuring for email languages 271
  configuring MCF_GETMAIL properties 267
Nortel Meridian 21
notes xxxi
notifications
  notifying clusters of changed parameters 86
  real-time event 49
    See Also REN servers
Notify Cluster page 86
NotifyQ
  demonstrating generic persistent events 125
  understanding 143
numAgentsAvailable 198, 304
numAgentsLoggedIn 198, 304
numberOfAbandoned 196, 304
numberOfLoggedIn 196, 304
numberOfQueued 197, 304
numEscalation 198, 304
numOfLines 179, 298
numOverflow 199, 304
numTaskAccepted 304
  GroupStatistics1 class 199
  UserStatistics1 class 261, 319
numTaskDone 199, 305
numTaskQueued 199
numTasksDone 261, 319
numTasksUnassigned 261, 319

O

Object 285
oldestTask 200, 305
onAccepted 286
onAccepted event 150
onAcceptingTask 150, 286
onAddressRegistered 237, 314
onAddressUnregistered 237, 314
onAlternating 212, 307
onAnswering 212, 307
onAttachingUD 212, 307
onBroadcast 230, 312
onBuddyRegistered 237, 314
onBuddyUnregistered 237, 314
onCallDataChanged 213, 307
onCancelingDnd 182, 298
onCancelingForward 182, 298
onCapabilitiesChanged 213, 251, 307, 316
onChatEnded 159, 289
onClearing 213, 307
onClosed 238, 314
onCompleting 213, 307
onConferencing 213, 307
onCtiBusy 251, 316
onCtiClear 251, 316
onDequeueingTask 150, 286

onDialing 213, 307
onDnd 182, 298
onDndCanceled 298
onDown 228, 311
onDropped 307, 317
  Line class 214
  User class 251
onError 307, 314
  _Address class 147, 285
  Line class 214
  Session class 238
  User class 252, 317
onForwardCanceled 182, 298
onForwarded 183, 298
onForwarding 183, 298
onGrabbing 214, 307
onGroupRegistered 238, 314
onGroupUnregistered 238, 314
onHbLost 230, 312
onHbRecovered 230, 312
onHeld 214, 307
onHolding 214, 307
onHook 308
onInfo 314, 317
  Session class 238
  User class 252
onInitiatingChat 159, 289
onInService 230, 312
onLoggedIn 252, 317
onLoggedOut 252, 317
onLoggingIn 252, 317
onLoggingOut 253, 317
onMediaDisabled 253, 317
onMediaEnabled 253, 317
onNotify
  A2AChatAddress class 159, 289
  _UQAddress class 150, 286
onNotReady 253, 317
onOffHook 215, 308
onOnHook 215
onOpened 238, 314
onOutOfService 230, 312
onParking 215, 308
onPartyAdded 215, 308
onPartyChanged 215, 308
onPartyRemoved 215, 308
onPresenceChanged 253, 317
onReady 253, 317
onReconnecting 216, 308
onRegistered 254, 317
onRegistering 254, 317
onReleasing 216, 308
onRestart 231, 312
onRetrieving 216, 308
onRinging 216, 308
onSettingDnd 183, 298
onSettingNotReady 254, 317
onSettingPresence 254, 317
onSettingReady 254, 317
onSettingWorkNotReady 255, 317
onSettingWorkReady 255, 317
onStat 303, 314
  Group class 195, 303
  Task class 241
  User class 255, 317
onStat1
  Buddy class 165, 291
  Group class 195, 303
  User class 255
onStat1(
  User class 317
onStat2
  Buddy class 165, 291
  Group class 195, 303
  User class 255, 317
onState 165, 291
onTalking 216, 308
onTaskAdded 303
  Group class 195
  _UQAddress class 286
  __UQAddress class 150
onTaskRemoved
  Group class 195
  __UQAddress class 150
onTransfering 217, 308
onUnassigned 151
  _UQAddress class 286
onUnknown 255, 317
onUnregistered 256, 317
onUnregistering 256, 317
onUp 228, 311
onUpdatingCD 217, 308
onUserDataChanged 217, 308
onUserRegistered 239, 314
onUserUnregistered 239, 314
onWorkNotReady 256, 317
onWorkReady 256, 317
open 233, 313
openSession 225, 310
Outbound Call page 36

outbound call sample page 35
overflow
administering overflow tasks 111
checking for overflowed persistent tasks 86
specifying task overflow default timeout 80
Overflow Administration component (MCF_OVERFLOWL_CMP) 111
Overflow Tasks page 111
overview
MultiChannel Framework 1

P

park 209, 307
PeopleBooks
ordering xxviii
PeopleCode
sample page 281
using built-in functions 142
PeopleCode Sample page 281
PeopleCode, typographical conventions xxx
PeopleSoft application fundamentals xxvii
PeopleSoft CTI, *See* CTI
PeopleSoft Integration Broker, *See* Integration Broker
PeopleSoft MultiChannel API (PSMCAPI), *See* PSMCAPI
performance issues
clustering REN servers 51
recording agent state information 104
restarting MCF servers 69
setting CTI agent tracing 29
tuning MCF clusters 82
permission lists, REN server 52
personalization
chat 95
CTI pop-ups 31, 44
dialing phone numbers on pages 36
floating CTI console 31
Personalization page (CTI) 31
Personalize Chat page 95
phone books
CTI 30
CTI shared 22
physical queues
active/inactive 75
administering 99
assigning agents 90
balancing agents/tasks across 102
deleting 74
displaying number of unassigned tasks/logged agents 83
forwarding chats 117
moving 101
moving agents 100
scalability 69
selecting MCF clusters 75
understanding 67
See Also queues
viewing MCF cluster IDs 75
viewing physical queue IDs 75
viewing state summary 104
Physical Queues component (MCF_ACCPT_TASK_CMP) 99
ping tests, REN server 61, 71
pop-ups
configuring 32
notifying CTI agent of calls 44
selecting pop-up mode for task/chat acceptance windows 94
setting up Cisco 34
setting up Genesys 33
signon screens 26
windows 31
POP3, *See* Post Office Protocol 3 (POP3)
Post Office Protocol 3 (POP3)
calculating email size 273
setting the connector threshold 270
prerequisites xxvii
presences 153, 287
presenceText 261, 319
primary queue servers 71
printed documentation xxviii
priority 241, 315
PSADMIN
configuring REN servers 56
setting application server tracing 110
PSMC class
constructor 222, 310
fields 223, 310
See Also PSMC class fields
method 223
methods 310
PSMC class fields
renserver 223
servers 223
sessions 223

PSMC class methods
closeSession 224
getCallById 224
getLineById 224
openSession 225
start 225
stop 225
PSMC constructor 222, 310
PSMC object 310
PSMCAPI 12
components 14
configuring 14
CTI architecture 11
file locations 14
security 14
PT_CTI_AGENT component 27
PT_CTI_DEMOOUTB component 35, 129
PT_CTI_MISC component 16
PT_CTI_QUEUE component 27

Q

question
AppData class 162, 290
Chat class 172, 294
GenericTask class 189, 302
Queue Configuration component (PT_CTI_QUEUE) 27
Queue Configuration page 28
queue IDs
character requirements 74
logical 90
physical 90
viewing physical queue IDs 75
Queue page 74
queue server
state 86
Queue Server State component (MCF_QSERVSTATE_CMP) 102
Queue Server State page 103
queue servers
assigning tasks to agents 77
See Also tasks
changing for MCF clusters 71
configuring MCF clusters 68
creating 70
failover 67
logging the status 84
master/slave 67
See Also master and slave servers
optimizing performance 82
primary/backup within MCF clusters 71
scalability 69
setting maximum agent timeouts 85
specifying interval for clearing deleted tasks from memory 86
viewing the server state 103
Queue State Summary component (MCF_QUEUESTATE_CMP) 102
Queue State Summary page 104
queue statistics
monitoring 83
queue statistics, monitoring 139
queuedWaitTime 197, 304
queues
CTI, configuring 28
logical 67
See Also logical queues
MCF 73
See Also MCF queues
monitoring 139
physical 67
See Also physical queues
queue servers 67
See Also queue servers
task queues, refreshing 85
universal 5
queueTime 169, 293

R

real-time event notification (REN) servers, *See* REN servers
reason 221, 309
Reason class
constructor 226, 311
fields 226, 311
See Also Reason class fields
Reason class fields
code 226
desc 227
Reason constructor 226, 311
Reason object 311
reasonFlag 305
GroupStatistics1 class 199
UserStatistics1 class 262, 319
recentTask 201, 305
reconnect 210, 307
register 247, 316
registerAddress 233, 313

registerBuddy 234, 313
registered 194
registeredAddresses 244, 316
registerGroup 234, 313
registerUser 234, 313
related documentation xxviii
relativeQueueLoad 197, 304
release 210, 307
REN Cluster component (REN_CLUSTER_CMP) 58
REN Permissions page 52
REN server
  configuring in PSADMIN 56
  security 14
REN Server Cluster Members page 61
REN Server Cluster Owner page 61
REN Server Cluster page 59
REN Server component (REN_SERVER_CMP) 53
REN Server Configuration page 58
REN servers
  clustering 59
  configuring 53, 55, 69
  configuring cluster members 61
  configuring clusters 58
  configuring failover 49
  configuring MCF clusters 68
  configuring ownership 61
  configuring scalability 49
  configuring security 49, 51
  creating 55
  defining 58
  defining permission lists 52
  determining configuration options 53
  logging 84
  performing buffer/ping tests 60, 71
  understanding 49
  understanding clusters 51
REN_CLUSTER_CMP component 58
REN_SERVER_CMP component 53
renserver 223
renServer 310
RenServer class
  callback event methods 311
  constructor 227, 311
  event method 228
    See Also RenServer class callback event methods
  fields 227, 311
    See Also RenServer class fields
RenServer class callback event methods
  onDown 228
  onUp 228
RenServer class fields
  isRunning 227
  url 227
RenServer constructor 227, 311
RenServer object 311
retrieve 210, 307
reverse proxy servers
  configuring REN server security 51
  configuring REN servers with SSL 54
  setting REN server browser URLs 60

S

Sample Monitor - Agent States 137
Sample Monitor - Queue Statistics 139
sample pages
  customer chat 122
  email 273
  Email page 127
  Email window 128
  Generic Event page 125
  Generic Event window 126
  instant messaging 281
  instant messaging button 282
  Monitor Agents page 137
  Monitor Queues page 139
  PeopleCode 281
  Sample Monitor - Agent States 137
  Sample Monitor - Queue Statistics 139
  URL wizard 123
  working with 121
scalability
  configuring MCF 69
  configuring REN server clusters 58
  configuring REN servers 49
  queue servers 69
  understanding 65
Secure Socket Layer (SSL)
  configuring REN server clusters 51
  configuring REN server security 51
  configuring REN/reverse proxy servers 54
  using the MultiChannel Console 114
security
  accessing CTI 39
  applying CTI applet protection 16
  configuring REN servers 49, 51
  opening MultiChannel consoles 114

PSMCAPI 14
REN server 14
setting security zones for web/REN servers 57, 114
setting the authentication domain for REN servers 56
specifying internal/external URLs for MCF clusters 68
Server class
callback event methods 312
constructor 228, 312
event method 230
See Also Server class callback event methods
fields 229, 312
See Also Server class fields
Server class callback event methods
onBroadcast 230
onHbLost 230
onHbRecovered 230
onInService 230
onOutOfService 230
onRestart 231
Server class fields
id 229
info 229
state 229
type 229
Server constructor 228, 312
Server object 312
Server.TYPE.CTI 302
Server.TYPE_CTI 191
Server.TYPE_UQ 191, 302
serverId 232
serverID 313
servers 223, 310
application 6
See Also application servers
instant messaging 279
See Also instant messaging
mail 267
See Also mail servers
master/slave 67
See Also master and slave servers
MCF 6
See Also MCF servers
MCF log 68
See Also MCF log servers
queue 67
See Also queue servers
REN 6, 49
See Also REN servers
reverse proxy 51
See Also reverse proxy servers
web servers 49
See Also REN servers
service request masks 26
Session class
callback event methods 314
constructor 231, 312
event method 237
See Also Session class callback event methods
fields 231, 313
See Also Session class fields
method 233
methods 313
Session class callback event methods
onAddressRegistered 237
onAddressUnregistered 237
onBuddyRegistered 237
onBuddyUnregistered 237
onClosed 238
onError 238
onGroupRegistered 238
onGroupUnregistered 238
onInfo 238
onOpened 238
onUserRegistered 239
onUserUnregistered 239
Session class fields
addresses 231
buddies 232
groups 232
id 232
serverId 232
state 232
user 232
Session class methods
close 233
open 233
registerAddress 233
registerBuddy 234
registerGroup 234
registerUser 234
statPublish 235
unregisterAddress 235
unregisterBuddy 236
unregisterGroup 236
unregisterUser 236

Session constructor 231, 312
Session object 312
sessions 223, 310
setDnd 181, 298
setNotReady 248, 316
setPresence 248, 316
setPresenceUq 248, 316
setReady 249, 316
setWorkNotReady 316
setWorkReady 250, 316
Shared Phone Book page 22
Siemens Hicom 300 21
signon, single, *See* single signon
Simple Mail Transfer Protocol (SMTP) 271
single signon
  accessing REN servers 51
  setting the default URL 26
  supporting on CTI 35
slave servers, *See* master and slave servers
SMTP 271
SSL, *See* Secure Socket Layer (SSL)
ST_LOGGEDIN 153, 287
ST_LOGGEDOUT 153, 287
ST_NOTREADY 153, 287
ST_READY 153, 287
ST_UNKNOWN 153, 287
ST_WORKNOTREADY 153, 287
ST_WORKREADY 154, 287
start 225, 310
state
  Line class 202, 306
  Server class 229, 312
  Session class 232, 313
  UserStatistics1 class 262, 319
states 153, 287
static push URLs
  defining 76
  defining agent-specific 96
statistics
  Call class 167, 292
  Group class 194, 303
  _User class 154, 287
statistics, logging 84
statistics1
  Group class 194, 303
  _User class 154, 287
statistics2
  Group class 194, 303
  _User class 154, 287
statPublish 235, 313
status
  detecting instant messaging presence 282, 283
  dialing outbound calls 45
  logging for agents 83
  logging for queue servers 84
  switching Agent Ready status 45
  understanding hold status 44
  viewing for CTI agents 40
stop 225, 310
strData 162, 290
subject
  AppData class 162, 290
  Chat class 172, 294
  GenericTask class 189, 302
suggestions, submitting xxxii

T

talkTime 170, 293
Task class
  constructor 240, 314
  fields 240, 314
    See Also Task class fields
  Hierarchy 239
Task class fields
  caseid 240
  cost 240
  customerid 241
  group 241
  id 241
  onStat 241
  priority 241
  type 241
  urlAbs 242
  urlRel 242
Task Configuration page 77
Task constructor 240, 314
Task object 314
task queues, refreshing 85
Task.TYPE_A2ACHAT 191, 302
Task.TYPE_CHAT 191, 302
Task.TYPE_CTI 192, 302
Task.TYPE_EMAIL 192, 302
Task.TYPE_GENERIC 192, 302
tasks
  A2AChatAddress class 158, 289
  activating 114
  active/inactive physical queues 75
  administering escalation/overflow 111

checking for expired/overflowed persistent tasks 86
configuring 77
delivering/assigning 6
displaying number of unassigned tasks per queue 83
managing via MultiChannel Console 113
managing with MultiChannel Console 6
moving agents 100
moving queues associated with agents 101
refreshing the task queue 85
selecting acceptance mode in chat 94
selecting task assignment actions 97
setting agent skill level 90
specifying chat greeting interval 80
specifying default acceptance timeout 79
specifying default cost 78
specifying default escalation/overflow timeout 80
specifying default priority/agent skill level 78
specifying how many/often to read persistent tasks 84, 85
specifying interval for clearing deleted tasks from memory 86
specifying range for parameter values 80
tuning clusters 82
universal queue 5
_UQAddress class 286
using PeopleCode built-in functions 142
viewing the enqueued task count 80
Tasks field 148
taskTotalTimeInSystem 199, 305
teleset 43
terms 321
time zone offsets 266
timeElapsedOldestTask 201, 305
timeElapsedRecentTask 201, 305
timeIdle 263, 319
timeInCurrentState 262, 263, 319
timeNotReady 263, 319
timeouts
setting maximum agent timeouts for queue server logout 85
specifying for chats 80
specifying the default for task acceptance 79
task escalation/overflow 80, 111
timeSinceLoggedIn 262, 319
timeSinceLogin 263, 319
timeSinceStart 199, 305
totalTimeAvailable 263, 319
totalTimeUnavailable 264, 319
tracing
application servers 110
setting for agents 97
setting for CTI agents 29
transfer 211, 307
transfer conference setup masks 26
transferMute 211, 307
transferring callers 42
troubleshooting CTI errors 29
type
A2AChat class 156, 288
MediaType class 222, 310
Server class 229, 312
Task class 241, 315
typographical conventions xxx

U

unavailableDuration 161, 290
uniqueId 162, 290
uniqueueId 157, 288
universal queue 5
unregister 250
unRegister 316
UserData class 318
unregisterAddress 235, 313
unregisterBuddy 236, 313
unregisterGroup 236, 313
unregisterUser 236, 313
updateCallData 212, 307
UQ Address constructor 148
UQ Cluster (universal queue cluster) page 70
_UQAddress class
callback event methods 286
constructor 148, 286
event method 149
See Also _UQAddress class callback event methods
fields 148, 286
methods 148, 286
See Also _UQAddress methods

_UQAddress class callback event methods
onAccepted 150
onAcceptingTask 150
onDequeueingTask 150
onNotify 150
onTaskAdded 150
onTaskRemoved 150
onUnassigned 151
_UQAddress class constructor 148
_UQAddress class fields 148
_UQAddress methods
acceptTask 149
dequeueTask 149
_UQAddress object 286
_UQAddress() constructor 286
url 290, 311
AppData class 163
RenServer class 227
URL Wizard page 123
urlAbs 242, 315
urlRel 242, 315
URLs
defining agent-specific 96
defining static push 76
generating when agents accept tasks 121
grabbing 123
REN server cluster 51
sending/grabbing via chat 117
specifying internal/external URLs for MCF clusters 68
using the URL wizard 123
viewing details for MCF clusters 81
user 309
MCEvent class 221
Session class 232, 313
User class
callback event methods 316
constructor 242, 315
event method 251
See Also User class callback event methods
fields 243, 315
See Also User class fields
method 244
methods 316
_User class 152
constructor 151, 287
fields 152, 287
User class callback event methods
onCapabilitiesChanged 251
onCtiBusy 251
onCtiClear 251
onDropped 251
onError 252
onInfo 252
onLoggedIn 252
onLoggedOut 252
onLoggingIn 252
onLoggingOut 253
onMediaDisabled 253
onMediaEnabled 253
onNotReady 253
onPresenceChanged 253
onReady 253
onRegistered 254
onRegistering 254
onSettingNotReady 254
onSettingPresence 254
onSettingReady 254
onSettingWorkNotReady 255
onSettingWorkReady 255
onStat 255
onStat1 255
onStat2 255
onUnknown 255
onUnregistered 256
onUnregistering 256
onWorkNotReady 256
onWorkReady 256
_User class field
ST_LOGGEDIN 153
states 153
User class fields
addresses 243
agentPassword 244
language 244
registeredAddresses 244
_User class fields
agentID 152
caps 152
id 152
name 152
presences 153
ST_LOGGEDOUT 153
ST_NOTREADY 153
ST_READY 153
ST_UNKNOWN 153
ST_WORKNOTREADY 153
ST_WORKREADY 154

statistics 154
statistics1 154
statistics2 154
User class methods
disableMedia 244
enableMedia 245
enableMediaregister 247
enableMediasetNotReady 248
isMediaEnabled 245
login 245
loginUq 246
logout 246
logoutUq 247
setPresence 248
setPresenceUq 248
setReady 249
setWorkNotReady 249
setWorkReady 250
unregister 250
User constructor 242, 315
_User constructor 151
User object 315
_User object 287
_User() constructor 287
UserCaps class
constructor 256, 257
UserCaps class fields
canLogin 257
canLogout 257
canRefreshState 257
canSetNotReady 257
canSetPresence 258
canSetReady 258
canSetWorkNotReady 258
canSetWorkReady 258
UserCaps constructor 256
userData 292
UserData class
constructor 258, 318
fields 259, 260
method 259
methods 318
UserData class field constant 259
UserData constructor 258, 318
UserData object 318
userId 163, 290
username 163, 290
UserStatistics1 class
constructor 260, 318
fields 318
UserStatistics1 class fields
availableCost 260
ctiBusy 260
currentQueue 261
mostRecentTaskData 261
mostRecentTaskId 261
numTaskAccepted 261
numTasksDone 261
numTasksUnassigned 261
presenceText 261
reasonFlag 262
state 262
timeInCurrentState 262
timeSinceLoggedIn 262
UserStatistics1 constructor 260, 318
UserStatistics1 object 318
UserStatistics2 class
constructor 262, 319
fields 263, 319
UserStatistics2 class fields
currentQueue 263
timeIdle 263
timeInCurrentState 263
timeNotReady 263
timeSinceLogin 263
totalTimeAvailable 263
totalTimeUnavailable 264
UserStatistics2 constructor 262, 319
UserStatistics2 object 319

V

visual cues xxxi
voice channels 4
See Also channels

W

waitDuration 161, 290
warnings xxxi
web servers, *See* REN servers
Window Config (configuration) page 94
windows
configuring for agents 94
configuring pop-ups 32
personalizing pop-ups 31
setting up Cisco for pop-ups 34
setting up Genesys for pop-ups 33
wizUrl 163, 290
workload
queue server scalability 69

setting maximum for agents 91
specifying for agents 78
See Also tasks
universal queue 5
wrap-up mode 45
wrap-up time, Genesys 46

Y

Yahoo! Messenger
configuring servers 279
detecting presence 279
setting the server IP address 280
updating presence 282