



Guidewire PolicyCenter™

Contact Management Guide

Release 10.0.0

©2001-2018 Guidewire Software, Inc.

For information about Guidewire trademarks, visit <http://guidewire.com/legal-notices>.

Guidewire Proprietary & Confidential — DO NOT DISTRIBUTE

Product Name: Guidewire ContactManager

Product Release: 10.0.0

Document Name: Contact Management Guide

Document Revision: 25-September-2018

Contents

About ContactManager documentation	19
Conventions in this document	20
Support	21
1 New and changed features in ContactManager	23
New and changed features in ContactManager 10.0.0	23
Changes in ContactManager 10.0.0	23
ContactManager 10.0 compatibility with 9.0.x core applications	23
ContactManager 10.0 web services, helper classes, WSDL files	25
Changes in ContactManager integration files	25
Changes to core application integration with ContactManager 10.0.0	26
Core application web service integration with ContactManager 10.0	26
ClaimCenter classes for ContactManager integration	26
PolicyCenter classes for ContactManager integration	27
BillingCenter classes for ContactManager integration	28
New and changed features in ContactManager 9.0.5	28
New and changed features in ContactManager 9.0.4	28
New and changed features in ContactManager 9.0.3	29
New and changed features in ContactManager 9.0.2	29
New and changed features in ContactManager 9.0.1	29
New and changed features in ContactManager 9.0.0	29
New features in ContactManager 9.0.0	29
Changes in ContactManager 9.0.0	29
ContactManager 9.0 compatibility with 8.0.x core applications	29
Apache Ant installation requirement removed	30
Changes in location of ab800 and ab801 web services, helper classes, and WSDL files	30
Changes in ContactManager file and package functionality	31
Changes in ABContact data model	32
Changes to core application integration with ContactManager 9.0.0	32
Core application web service integration with ContactManager	32
Changes to AddressBookUID and LinkID	32
ClaimCenter classes for ContactManager integration	33
PolicyCenter classes for ContactManager integration	33
BillingCenter classes for ContactManager integration	34
Maximum length of LinkID and AddressBookUID fields has been increased	34
New and changed features in ContactManager 8.0.7	35
New and changed features in ContactManager 8.0.6	35
New and changed features in ContactManager 8.0.5	35
New and changed features in ContactManager 8.0.4	35
New and changed features in ContactManager 8.0.3	36
Change to configuring the ClaimCenter exit point to ContactManager	36
New and changed features in ContactManager 8.0.2	36
Configuring the number of services specified in a search	36
Change to order of proximity based search results	36
New and changed features in ContactManager 8.0.1	36
Core applications can specify a unique id when creating a contact	37
Changes to contact linking and synchronizing APIs in ClaimCenter	37
Approving and editing a pending update or create	37

Finding duplicate contacts for a pending create	38
Merging and editing potential duplicate contacts	38
Changing the subtype of a Contact instance	38
Using an Address field to search for contacts	38
Changes to ab800 web services, helper classes, and WSDL files	39
Changes to core application integration with ContactManager	39
New and changed features in ContactManager 8.0.0	40
New features in ContactManager 8.0.0	40
Pending creates and pending changes to contacts	40
Changes in ContactManager 8.0.0	40
Changes to contact plugins, web services, and mapper classes	40
Changes to search classes	45
Changes to contact search functionality	46
Contact entity mapping classes and XML files	46
Changes to ClaimCenter contact synchronization	47
ContactManager 8.0 compatibility with core applications	47
2 Managing integrated contacts	49
Client Data Management	49
PolicyCenter role in managing client data	49
PolicyCenter linked addresses and side effects	50
ClaimCenter role in managing client data	50
BillingCenter role in managing client data	50
Vendor Data Management	51
Vendor contacts	51
Using and configuring vendor data management	51
Overview of ClaimCenter integration with ContactManager	51
Centralized vendor data management	51
Which contacts to store in ContactManager	52
Locally and centrally managed contacts	52
Deciding whether to use ContactManager for vendor management	53
3 Installing ContactManager	55
Installing ContactManager with QuickStart for development	55
Install ContactManager with QuickStart	55
Load sample data for ContactManager	56
4 Integrating ContactManager with Guidewire core applications.	59
Integrating ContactManager using QuickStart	59
Integrating ContactManager with ClaimCenter in QuickStart	59
Integrate ClaimCenter with ContactManager	61
Integrate ContactManager with ClaimCenter	62
Test the integration of ClaimCenter and ContactManager	64
Troubleshooting the ClaimCenter connection with ContactManager	65
Integrating ContactManager with PolicyCenter in QuickStart	66
Integrate PolicyCenter with ContactManager	67
Integrate ContactManager with PolicyCenter	69
Test the integration of PolicyCenter and ContactManager	71
Troubleshooting the PolicyCenter connection with ContactManager	72
Integrating ContactManager with BillingCenter in QuickStart	72
Integrate BillingCenter with ContactManager	73
Integrate ContactManager with BillingCenter	75
Test the integration of BillingCenter and ContactManager	77
Troubleshooting the BillingCenter connection with ContactManager	78
Configuring core application authentication with ContactManager	78
Configure ClaimCenter-to-ContactManager authentication	79

Configure PolicyCenter-to-ContactManager authentication	80
Configure BillingCenter-to-ContactManager authentication	81
Configuring ContactManager authentication with core applications	82
Overview of ContactManager authentication configuration.	82
Configure ContactManager-to-ClaimCenter authentication.	83
Configure ContactManager-to-PolicyCenter authentication	85
Configure ContactManager-to-BillingCenter authentication	87
5 Searching for contacts	91
Overview of contact search	91
ContactManager support for contact searches.	92
Local ContactManager search criteria	92
Defining minimum search criteria for a contact.	92
Effect of locale and country on minimum search criteria	93
Minimum search criteria error messages	93
ContactManager support for core application searches	94
Configuring Contact search criteria and search results in ContactManager.	94
Limiting the number of service elements specified in a Contact search	94
ClaimCenter support for contact searches	94
Adding the InterpreterSpecialty property to search	95
Add search support to the ContactManager user interface	95
Add search support in ContactManager for Guidewire core applications	96
Configure the address book search interface in ContactManager	97
Restart ContactManager and refresh web services in ClaimCenter	98
Add ContactManager search capability in ClaimCenter.	99
Configure the address book search interface in ClaimCenter	100
Test the InterpreterSpecialty search extensions.	101
Adding an address field to Contact search.	101
Add a normal address field to search.	101
Add a denormalized address field to search	102
Adding the service state property to search	103
Overview of adding search capability in ContactManager	103
Enable ServiceState support for the ContactManager search screens.	103
Add ServiceState search support in ContactManager for core applications.	105
Configure ContactManager search screens for ServiceState	105
Regenerate and refresh ContactManager web services.	106
Add ServiceState search capability in ClaimCenter.	106
Configure ClaimCenter search screens for ServiceState.	107
Test the ServiceState search extensions	108
Geocoding and proximity search for vendor contacts	108
Geocoding and batch processing	109
Configuring the geocoding work queues	109
BatchGeocode and GeocodeStatus properties in ContactManager.	110
Run Geocode batch processing manually in ClaimCenter	110
Run Geocode batch processing manually in ContactManager.	110
Configuring geocoding for ClaimCenter and ContactManager	110
Activate geocoding in Bing Maps.	111
Enable and use the Geocoding plugin	111
Set geocoding configuration parameters in ClaimCenter	112
Set geocoding configuration parameters in ContactManager	113
Schedule geocoding	114
Log geocoding information	115
Stop and restart ClaimCenter and ContactManager.	115
Proximity search example for vendor contacts.	116
Request distanced-based proximity search	116

PolicyCenter support for Contact searches	117
BillingCenter support for Contact searches	118
6 Securing access to contact information	121
ContactManager contact security	121
ContactManager user roles	122
ContactManager contact subtype and tag permissions	124
ContactManager contact and tag permission check expressions	125
Permission check expression parameters in ContactManager	125
Viewing permissions in the ContactManager Security Dictionary	126
Build and view the ContactManager Security Dictionary	127
Contact search security configuration parameters in ContactManager	127
Configuring ContactManager contact security	127
Create permissions for an ABCContact subtype in ContactManager	127
Creating client tag permissions in ContactManager	129
Create new claim party and vendor permissions and associated role	130
PolicyCenter contact security	132
BillingCenter contact security	132
BillingCenter contact-related permissions	132
BillingCenter contact and tag permission check expressions	133
Viewing permissions in the BillingCenter Security Dictionary	133
Build and view the BillingCenter Security Dictionary	134
ClaimCenter contact security	134
ClaimCenter contact roles	134
ClaimCenter contact subtype and tag permissions	134
ClaimCenter contact and tag permission check expressions	135
Viewing permissions in the ClaimCenter Security Dictionary	136
Build and view the ClaimCenter Security Dictionary	137
Contact permission search configuration parameters in ClaimCenter	137
Configuring ClaimCenter contact security	137
Create contact permissions for a subtype in ClaimCenter	138
Create claim party and vendor tag permissions in ClaimCenter	140
7 Extending the contact data model	143
Overview of contact entities	143
ABCContact data model	143
ABCContact entity hierarchy	143
ABCContact entity relationships	144
Contact data model	145
Contact entity hierarchy	145
Contact entity relationships	146
ContactManager and core application contact entity hierarchies	147
General guidelines for extending contacts	149
Deciding whether to create a subtype	149
Refresh applications after contact data model changes	150
Limitations on configuring contact entity extensions	150
Working with contact mapping files	150
ContactManager mapping	151
Core application mapping	151
Extending the client data model	152
Extending the vendor contact data model	152
Adding a field to a contact subtype	152
Add a field to the Doctor subtype in ClaimCenter	152
Add the new field to the ContactMapper class in ClaimCenter	153
Add the BoardCertified field to the Address Book user interface	154
Add the BoardCertified field to the claim contacts user interface	154

Add a field to the ABDoctor subtype in ContactManager	155
Add the new field to the ContactMapper class in ContactManager	156
Add the BoardCertified field to the ContactManager user interface	156
Restart both applications and test the new field	157
Adding a vendor contact subtype	158
Add a new Contact subtype to ClaimCenter	158
Add the new subtype to the ContactMapper class in ClaimCenter	160
Map the new subtype names in ClaimCenter	160
Add display keys for the new subtype to ClaimCenter	161
Add the subtype to the ClaimCenter New Contact menu	161
Add the new subtype to the Address Book input sets	162
Add the new contact to the shared contacts detail view	164
Add a new ABContact subtype to ContactManager	165
Add the new subtype to the ContactMapper class in ContactManager	167
Add display keys for the new subtype to ContactManager	167
Add the new subtype to the ContactManager actions menu	168
Add ABInterpreter fields to ContactManager modal input sets	168
Add the new subtype to the ContactManager contact picker menus	169
Restart both applications and test the new subtypes	170
Extending contacts with an array	171
Create a service state entity in ContactManager	171
Add the new array to the ABContact entity	174
Map the entity and array extensions in ContactManager	175
Add support for service states to the ContactManager user interface	176
Create a service state entity in ClaimCenter	177
Add the new array to the Contact entity in ClaimCenter	179
Map the new entity and array extensions in ClaimCenter	180
Add support for service state searches to the Address Book	181
Enable addition of service states to a company on a claim	182
Test service state entity and array extensions	184
Changing the subtype of a Contact instance	185
Changing subtype with a command-prompt utility	185
Restrictions on changing the subtype of a contact instance	186
Contact instance location and changing subtype	186
Subtype changes for client contacts	187
Contact subtype field differences and changing subtypes	187
Changing the subtype of a contact instance	188
Change the subtype of an ABContact instance in ContactManager	188
Change the subtype of a Contact instance in ClaimCenter	189
Troubleshooting the change subtype command-prompt utility	190
8 Contact tags	191
Applications and contact tags	191
Add a new contact tag	192
ClaimCenter contact tag generation	192
PolicyCenter contact tag generation	192
BillingCenter contact tag generation	192
Contact tag-based security	192
Contact tag permissions	192
Contact tag permission check expressions	193
9 Linking and synchronizing contacts	195
Linking a contact	195
Creating and linking a contact	195
PolicyCenter contact creation with external unique IDs	196
ContactManager support for handling external unique IDs	196

PolicyCenter support for creating external unique IDs.	197
Linking in ClaimCenter	197
Linking in PolicyCenter	198
Linking in BillingCenter.	199
Find duplicates behavior.	199
Synchronizing with ContactManager	200
ContactManager plugins for broadcasting of contact changes	200
Synchronizing PolicyCenter and ContactManager contacts.	201
Exceptions to PolicyCenter synchronization updates.	201
Configuring PolicyCenter external contact synchronization	201
Synchronizing PolicyCenter contact fields.	202
Synchronizing BillingCenter and ContactManager contacts	202
Exceptions to BillingCenter synchronization updates	202
Configuring BillingCenter external contact synchronization	202
Synchronizing BillingCenter contact fields	203
Synchronizing ClaimCenter and ContactManager contacts	203
Configuring automatic synchronization with ClaimCenter.	204
ClaimCenter synchronizing support	204
Initiating a manual synchronization from ClaimCenter	206
Synchronizing ClaimCenter contact fields	207
Excluding properties from contact synchronization.	207
Including and excluding contact relationships in synchronization	207
10 ContactManager rules	209
ContactManager rule sets	209
EventMessage rule set category.	209
Messaging events	210
Link a message event to a message destination.	210
Message destinations and message plugins	210
Generating messages.	211
Modifying entity data in event fired rules and messaging plugins	211
Preupdate rule set category	211
Entities that trigger preupdate rules.	212
Creating preupdate rules for custom entities.	212
ABContactContact preupdate rule set	213
ABContact preupdate rule set.	213
Address preupdate rule set.	213
PendingContactChangePreupdate rule set	213
Validation rule set category.	213
Overview of ContactManager preupdate and validation.	213
Overview of ContactManager validation	214
Validation graph in ContactManager	214
Top-level ContactManager entities that trigger validation	215
11 Configuring personal data destruction in ContactManager.	217
Overview of data destruction	217
Encapsulation of business logic for retention and destruction	217
Notification of data protection officer on errors or conflicts	218
Wide-swath data destruction	218
Individual-entity data destruction	218
Integration with Guidewire core applications	219
Integration with other systems.	219
Notification of downstream systems	219
Data destruction configuration parameters.	219
Data destruction web service	219
PersonalDataDestructionAPI web service methods	220

Lifecycle of a personal data destruction request	221
Personal data destruction request entities	221
Example of a request made with AddressBookUID	222
Example of a request made with PublicID	222
Typelists for status of personal destruction workflow	222
Work queues used in personal data destruction	223
DestroyContactForPersonalData work queue	223
NotifyExternalSystemForPersonalData work queue	223
RemoveOldContactDestructionRequest work queue	224
PersonalDataDestructionController class	224
Data model configuration for data destruction	224
DestructionRootPinnable delegate	225
Root of entity graph for personal data destruction	225
Do Not Destroy flag	225
Do Not Process flag	225
Display keys for data destruction messages	225
Obfuscatable delegate	225
Obfuscator interface	226
Marking entity fields for obfuscation	226
ContactManager entity domain graph	227
Overview of the ContactManager entity domain graph	227
ContactManager ABContact entity domain graph	228
ABContact domain graph and related entities	228
Table ab_purgedrootinfo	230
Extensions of ABContact and other entities and the domain graph	230
ContactManager Data Protection Officer	230
Data Protection Officer permissions	230
Notifying the Data Protection Officer	231
Data destruction purge configuration	231
Defining the Destroyer	231
PersonalDataPurge event	232
Specifying which objects in the graph can be destroyed	233
PersonalDataDestruction plugin interface	233
Personal data destruction plugin implementation classes	233
Data obfuscation in ContactManager	234
Implementing the Obfuscatable delegate in an entity	235
Implementing the Obfuscator interface in an entity	235
Personal data obfuscation classes	236
Personal data obfuscation class hierarchy	236
PersonalDataObfuscator	236
UnsupportedObfuscator	237
PersonalDataObfuscatorUtil	237
12 Working directly in ContactManager	239
Logging in to ContactManager	239
ContactManager login requirements	239
Log in to ContactManager	239
Change your password in ContactManager	240
Change your user preferences in ContactManager	240
Preferences worksheet	240
Selecting international settings in ContactManager	241
ContactManager user interface	241
Contacts tab	242
Administration tab	242
Internal tools and server tools tabs	243

International settings in ContactManager	243
Changing the screen layout	244
Importing and exporting administrative data	245
Import administrative data	245
Exporting data in the administration tab	245
Importing and exporting either with Gosu or from the command prompt	246
Managing contact data	246
Detecting and merging duplicate contacts	247
Configuring Duplicate Contacts Finder batch processing	247
Set the DuplicateContactsEarliestModificationDate configuration parameter	248
Running the Duplicate Contacts Finder work queue manually	248
Setting Duplicate Contacts Finder to run automatically	249
Merge duplicate contacts	251
Merging contacts and notifying core applications	253
Review pending changes to contacts	253
Pending changes screen cache	255
Working with work queues and batch processes	255
Start work queues and batch processes	255
Batch Process Info screen	255
Improve query performance for large batch jobs	256
13 ContactManager integration reference	259
Overview of ContactManager integration	259
ContactManager entities	260
ContactManager link IDs and comparison to other IDs	260
ContactManager web services	261
Web services provided by ContactManager	261
Support classes for ContactManager web services	263
ABContactAPI web service	264
ABContactAPI and typelists	264
ABContactAPI methods	264
Retrieving contacts and their relationships	268
ABVendorEvaluationAPI web service	268
ContactManager messaging events	269
ContactManager messaging events by entity	269
ABContact message safe ordering	271
ABContact message resynchronization	271
ABClientAPI interface	272
Method signatures of the ABClientAPI interface	272
Deleting contacts in ClaimCenter	273
ClaimCenter ContactAPI web service methods for removing contacts	273
ClaimCenter classes for removing contacts	274
Implementation details for retiring contacts in ClaimCenter	274
ContactMapper class	275
ContactManager ContactMapper class	276
Mapping fields of a ContactManager contact	276
Mapping externally specified unique IDs of a ContactManager contact	277
Mapping foreign keys of a ContactManager contact	277
Mapping array references of a ContactManager contact	278
Special handling in ContactManager for addresses	278
Core application ContactMapper class	278
Mapping fields of a core application contact	279
Mapping foreign keys of a core application contact	280
Mapping array references of a core application contact	280
PolicyCenter mapping of externally specified unique IDs	281

Special handling in ContactMapper for core applications	281
Excluding contact fields from ClaimCenter contact synchronization	282
Excluding contact fields from being saved in the ClaimCenter database	282
Overview of ContactManager plugins	283
Register a plugin implementation in ContactManager	283
ContactManager plugins	284
IFindDuplicatesPlugin plugin interface	289
Duplicate finder classes	290
CompanyDuplicateFinder class	290
PersonDuplicateFinder class	291
PersonVendorDuplicateFinder class	292
PlaceDuplicateFinder class	292
UserDuplicateFinder class	293
Query builder classes	293
ContactQueryBuilder class	293
CompanyQueryBuilder class	295
PlaceQueryBuilder class	295
PersonQueryBuilderBase class	295
PersonQueryBuilder class	296
UserContactQueryBuilder class	296
ValidateABContactCreationPlugin plugin interface	297
Contact minimum creation requirements	297
Verify Minimum Criteria work queue	297
Configure the TaxID contact creation requirement	298
Code example for creating ABContact and ABCompany contacts	298
Configuring validation error messages for contact creation	298
Testing clock plugin interface	299
14 Release note archive	301
Guidewire ContactManager 9.0.5 Release Notes	301
Release information	301
Release number	302
Installing this release	302
Upgrade information	302
Support	302
Major issues and changes	302
Base PCF file changes	302
Base rule changes	303
Changes in this release provided in Upgrade Diff report	303
Improvements and resolved issues	303
ContactManager 905 improvements and resolved Issues	303
Platform improvements and resolved issues	303
Known issues and limitations	307
ContactManager known issues	307
Studio/Platform known issues	308
Guidewire ContactManager 9.0.4 release notes	310
Release information for ContactManager 9.0.4	310
Installing ContactManager 904	310
Upgrade information for ContactManager 9.0.4	311
Support	311
Major issues and changes for ContactManager 9.0.4	311
Base PCF file changes for ContactManager 9.0.4	311
Base rule changes for ContactManager 904	311
Changes in this release provided in Upgrade Diff report	311
Improvements and resolved issues for ContactManager 9.0.4	311

ContactManager 904 improvements and resolved issues	312
Platform improvements and general issues for ContactManager 904	312
Known issues and limitations for ContactManager 9.0.4	314
ContactManager 9.0.4 known issues	315
Studio/Platform known issues for ContactManager 9.0.4	315
Guidewire ContactManager 9.0.3 release notes	317
Release information for ContactManager 9.0.3	317
Installing ContactManager 9.0.3	317
Upgrade information for ContactManager 9.0.3	317
Support	317
Major issues and changes for ContactManager 9.0.3	318
New Studio options for managing application upgrades (IDE-3811, IDE-3960)	318
Improve build performance by preprocessing XML schema files (PL-36353)	318
Base PCF file changes for ContactManager 9.0.3	318
Base rule changes for ContactManager 9.0.3	318
Changes in this release provided in Upgrade Diff report	318
Improvements and resolved issues for ContactManager 9.0.3	318
Platform improvements and general issues for ContactManager 9.0.3	318
Known issues and limitations for ContactManager 9.0.3	320
ContactManager 9.0.3 known issues	320
Studio/Platform issues for ContactManager 9.0.3	320
Guidewire ContactManager 9.0.2 release notes	321
Release Information for ContactManager 9.0.2	321
Installing ContactManager 9.0.2	322
Upgrade Information for ContactManager 9.0.2	322
Support	322
Major issues and changes for ContactManager 9.0.2	322
New and changed features for ContactManager 9.0.2	322
Free text search is supported on WebLogic	322
Base PCF file changes for ContactManager 9.0.2	323
Base rule changes for ContactManager 9.0.2	323
Changes in this release provided in Upgrade Diff report	323
Improvements and resolved issues for ContactManager 9.0.2	323
ContactManager 9.0.2 improvements and resolved issues	323
Platform improvements and resolved issues for ContactManager 9.0.2	323
Known issues and limitations for ContactManager 9.0.2	325
ContactManager 9.0.2 known issues	325
Studio/Platform known issues for ContactManager 9.0.2	326
Guidewire ContactManager 9.0.1 release notes	327
Release Information for ContactManager 9.0.1	327
Installing ContactManager release 9.0.1	327
Supported Java versions for ContactManager 9.0.1	328
Install the latest Guidewire Studio update for ContactManager 9.0.1	328
Upgrade Information for ContactManager 9.0.1	328
Support	328
Major issues and changes for ContactManager 9.0.1	328
New and changed features for ContactManager 9.0.1	328
Base PCF file changes for ContactManager 9.0.1	329
Base rule changes for ContactManager 9.0.1	329
Changes in this release provided in Upgrade Diff report	329
Improvements and resolved issues for ContactManager 9.0.1	329
ContactManager 9.0.1 improvements and resolved issues	329
Platform improvements and resolved issues for ContactManager 9.0.1	330
Known issues and limitations for ContactManager 9.0.1	332
ContactManager 9.0.1 known issues	332

Studio/Platform known issues for ContactManager 9.0.1	332
Guidewire ContactManager 9.0.0 release notes	334
Release Information for ContactManager 9.0.0	334
Installing ContactManager 9.0.0	335
Supported Java versions for ContactManager 9.0.0	335
Install the latest Guidewire Studio update	335
Upgrade Information for ContactManager 9.0.0	335
Support	335
Major issues and changes for ContactManager 9.0.0	336
New and changed features for ContactManager 9.0.0	336
Oracle support planned for ContactManager 9.0.0	336
Base PCF file changes for ContactManager 9.0.0	336
Base rule changes for ContactManager 9.0.0	336
Changes in this release provided in Upgrade Diff report	336
Improvements and resolved issues for ContactManager 9.0.0	336
ContactManager 9.0.0 improvements and resolved issues	337
Known issues and limitations for ContactManager 9.0.0	337
ContactManager 9.0.0 known issues	337
Studio/Platform known issues for ContactManager 9.0.0	338
Guidewire ContactManager 8.0.7 release notes	339
Release Information for ContactManager 8.0.7	339
Installing ContactManager 8.0.7	339
Upgrade Information for ContactManager 8.0.7	340
Support	340
Major issues and changes for ContactManager 8.0.7	340
New and changed features for ContactManager 8.0.7	340
Base PCF file changes for ContactManager 8.0.7	340
Base rule changes for ContactManager 8.0.7	341
Changes in this release provided in Upgrade Diff report	341
Improvements and resolved issues for ContactManager 8.0.7	341
Platform improvements, resolved issues for ContactManager 8.0.7	341
Known issues and limitations for ContactManager 8.0.7	342
ContactManager 8.0.7 known issues	342
Studio/Platform known issues for ContactManager 8.0.7	343
Guidewire ContactManager 8.0.6 release notes	344
Release Information for ContactManager 8.0.6	344
Installing ContactManager 8.0.6	345
Upgrade Information for ContactManager 8.0.6	345
Support	345
Major issues and changes for ContactManager 8.0.6	345
New and changed features for ContactManager 8.0.6	346
Base PCF file changes for ContactManager 8.0.6	346
Base rule changes for ContactManager 8.0.6	346
Changes in this release provided in Upgrade Diff report	346
Improvements and resolved issues for ContactManager 8.0.6	346
ContactManager 8.0.6 improvements and resolved issues	346
Platform improvements, resolved issues for ContactManager 8.0.6	346
Known issues and limitations for ContactManager 8.0.6	349
ContactManager 8.0.6 known issues	350
Studio/Platform known issues for ContactManager 8.0.6	350
Guidewire ContactManager 8.0.5 release notes	352
Release information for ContactManager 8.0.5	352
Installing ContactManager 8.0.5	352
Upgrade information for ContactManager 8.0.5	352
Support	353

Major issues and changes for ContactManager 8.0.5	353
New and changed features for ContactManager 8.0.5	353
Base PCF file changes for ContactManager 8.0.5	353
Base rule changes for ContactManager 8.0.5	353
Changes in this release provided in Upgrade Diff report	353
Improvements and resolved issues for ContactManager 8.0.5	353
ContactManager 8.0.5 improvements and resolved issues	353
Platform improvements and resolved issues for ContactManager 8.0.5	354
Known issues and limitations for ContactManager 8.0.5	357
ContactManager 8.0.5 known issues	357
Studio/Platform known issues for ContactManager 8.0.5	358
Guidewire ContactManager 8.0.4 release notes	359
Release Information for ContactManager 8.0.4	359
Installing ContactManager 8.0.4	359
Upgrade Information for ContactManager 8.0.4	360
Support	360
Major issues and changes for ContactManager 8.0.4	360
New and changed features for ContactManager 8.0.4	360
Base PCF file changes for ContactManager 8.0.4	360
Base rule changes for ContactManager 8.0.4	361
Changes in this release provided in Upgrade Diff report	361
Improvements and resolved issues for ContactManager 8.0.4	361
ContactManager 8.0.4 improvements and resolved issues	361
Platform improvements and resolved issues for ContactManager 8.0.4	361
Known issues and limitations for ContactManager 8.0.4	364
ContactManager 8.0.4 known issues	364
Studio/Platform known issues for ContactManager 8.0.4	365
Guidewire ContactManager 8.0.3 release notes	366
Release Information for ContactManager 8.0.3	366
Installing ContactManager 8.0.3	366
Upgrade Information for ContactManager 8.0.3	367
Support	367
Major issues and changes for ContactManager 8.0.3	367
New and changed features for ContactManager 8.0.3	367
Improvements to List View Navigation (PL-26518)	368
Base PCF file changes for ContactManager 8.0.3	368
Base rule changes for ContactManager 8.0.3	368
Changes in this release provided in Upgrade Diff report	368
Improvements and resolved issues for ContactManager 8.0.3	368
ContactManager 8.0.3 improvements and resolved issues	368
Platform improvements and resolved issues for ContactManager 8.0.3	369
Documentation improvements and resolved issues for ContactManager 8.0.3	374
Known issues and limitations for ContactManager 8.0.3	375
ContactManager 8.0.3 known issues	375
Studio/Platform known issues for ContactManager 8.0.3	375
Guidewire ContactManager 8.0.2 release notes	377
Release Information for ContactManager 8.0.2	377
Version number	378
Installing ContactManager 8.0.2	378
Upgrade Information for ContactManager 8.0.2	378
Support	378
Major issues and changes for ContactManager 8.0.2	379
Base PCF file changes for ContactManager 8.0.2	379
Base rule changes for ContactManager 8.0.2	379
Changes in this release provided in Upgrade Diff report	379

Improvements and resolved issues for ContactManager 8.0.2	379
ContactManager 8.0.2 improvements and resolved issues	379
Platform improvements and resolved issues for ContactManager 8.0.2	380
Known issues and limitations for ContactManager 8.0.2	385
ContactManager 8.0.2 known issues	385
Platform/Studio known issues for ContactManager 8.0.2	385
Guidewire ContactManager 8.0.1 release notes	388
Release Information for ContactManager 8.0.1	388
Installing ContactManager 8.0.1	388
Support	389
Issues and major changes for ContactManager 8.0.1	389
Base PCF file changes for ContactManager 8.0.1	389
Rule changes for ContactManager 8.0.1	389
Changes in this release provided in Upgrade Diff report	389
Improvements and general issues for ContactManager 8.0.1	390
Known issues and limitations for ContactManager 8.0.1	397
ContactManager 8.0.1 known issues	397
Platform/Studio known issues for ContactManager 8.0.1	398
Guidewire ContactManager 8.0.0 release notes	400
Release Information for ContactManager 8.0.0	400
Installing ContactManager 8.0.0	401
Support	401
Issues and major changes for ContactManager 8.0.0	401
Base PCF file changes for ContactManager 8.0.0	401
Rules changes for ContactManager 8.0.0	401
Changes in this release provided in Upgrade Diff report	401
Known issues and limitations for ContactManager 8.0.0	402
ContactManager known issues for release 8.0.0	402
Platform/Studio known issues for ContactManager 8.0.0	403
Guidewire ContactManager 7.0.7 release notes	405
Release Information for ContactManager 7.0.7	405
Installing ContactManager 7.0.7	406
Support	406
Major issues and changes for ContactManager 7.0.7	406
Allowing Core Applications to Specify the LinkId for a New Contact (CTC-3078)	406
Internet Explorer 11 is now supported (PL-30386)	407
Base PCF file changes in ContactManager 7.0.7	407
Base rule changes in ContactManager 7.0.7	407
Changes in this release provided in Upgrade Diff report	407
Improvements and resolved issues for ContactManager 7.0.7	407
ContactManager 7.0.7 improvements and resolved issues	407
Platform improvements and resolved issues for ContactManager 7.0.7	408
Known issues and limitations for ContactManager 7.0.7	409
ContactManager 7.0.7 known issues	410
Studio/Platform known issues for ContactManager 7.0.7	410
Guidewire ContactManager 7.0.6 release notes	412
Release Information for ContactManager 7.0.6	412
Installing ContactManager 7.0.6	412
Support	412
Issues and major changes for ContactManager 7.0.6	413
Base PCF file changes for ContactManager 7.0.6	413
Changes in this release provided in Upgrade Diff report	413
Improvements and general issues for ContactManager 7.0.6	413
Known issues and limitations for ContactManager 7.0.6	414
ContactManager 7.0.6 known issues	414

Studio/Platform known issues for ContactManager 7.0.6	414
Guidewire ContactManager 7.0.5 release notes	416
Release Information for ContactManager 7.0.5	416
Installing ContactManager 7.0.5	417
Support	417
Issues and major changes for ContactManager 7.0.5	417
Base PCF file changes for ContactManager 7.0.5	417
Rules changes for ContactManager 7.0.5	418
Changes in this release provided in Upgrade Diff report	418
Improvements and general issues for ContactManager 7.0.5	418
Known issues and limitations for ContactManager 7.0.5	420
ContactManager known issues for release 7.0.5	420
Studio/Platform known issues for ContactManager 7.0.5	421
Guidewire ContactManager 7.0.4 release notes	423
Release Information for ContactManager 7.0.4	423
Installing ContactManager 7.0.4	423
Support	424
Issues and major changes for ContactManager 7.0.4	424
Base PCF file changes for ContactManager 7.0.4	424
Rules changes for ContactManager 7.0.4	424
Changes in this release provided in Upgrade Diff report	424
Improvements and general issues for ContactManager 7.0.4	424
Known issues and limitations for ContactManager 7.0.4	426
ContactManager known issues for release 7.0.4	426
Studio/Platform Known Issues for ContactManager 7.0.4	427
Guidewire ContactManager 7.0.3 release notes	429
Release Information for ContactManager 7.0.3	429
Installing ContactManager 7.0.3	429
Support	429
Issues and major changes for ContactManager 7.0.3	429
Base PCF file changes for ContactManager 7.0.3	430
Rules changes for ContactManager 7.0.3	430
Changes in this release provided in Upgrade Diff report	430
Improvements and general issues for ContactManager 7.0.3	430
Known issues and limitations for ContactManager 7.0.3	433
ContactManager known issues for release 7.0.3	433
Studio/Platform Known Issues for ContactManager 7.0.3	434
Guidewire ContactManager 7.0.2 release notes	436
Release notes update: 04-June-2012.	436
Release Information for ContactManager 7.0.2	436
Installing ContactManager 7.0.2	436
Support	437
Issues and major changes for ContactManager 7.0.2	437
Changes in this release provided in Upgrade Diff report	437
Improvements and general issues for ContactManager 7.0.2	437
Known issues and limitations for ContactManager 7.0.2	444
ContactManager known issues for release 7.0.2	444
Studio/Platform known issues for ContactManager 7.0.2	444
Guidewire ContactManager 7.0.1 release notes	447
Release Information for ContactManager 7.0.1	447
Installing ContactManager 7.0.1	447
Support	447
Issues and major changes for ContactManager 7.0.1	447
ContactManager Integration with Core Applications.	448
Geocoding Using Bing and MapPoint (PL-16708)	448

New web services APIs return current AddressBookUID for merged contact (CTC-588)	448
Provide web service for validating contacts for creation (CTC-603)	448
Make ValidateABContactCreationPlugin return the error message (CTC-611)	448
Make ValidateABContactSearchCriteriaPlugin return the error message (CTC-612)	449
Changes in this release provided in Upgrade Diff report	449
Improvements and general issues for ContactManager 7.0.1	449
Known issues and limitations for ContactManager 7.0.1	457
ContactManager known issues	457
Studio/Platform known issues	457

About ContactManager documentation

The following table lists the documents in PolicyCenter documentation:

Document	Purpose
<i>InsuranceSuite Guide</i>	If you are new to Guidewire InsuranceSuite applications, read the <i>InsuranceSuite Guide</i> for information on the architecture of Guidewire InsuranceSuite and application integrations. The intended readers are everyone who works with Guidewire applications.
<i>Application Guide</i>	If you are new to PolicyCenter or want to understand a feature, read the <i>Application Guide</i> . This guide describes features from a business perspective and provides links to other books as needed. The intended readers are everyone who works with PolicyCenter.
<i>Database Upgrade Guide</i>	Describes the overall ContactManager upgrade process, and describes how to upgrade your ContactManager database from a previous major version. The intended readers are system administrators and implementation engineers who must merge base application changes into existing ContactManager application extensions and integrations.
<i>Configuration Upgrade Guide</i>	Describes the overall ContactManager upgrade process, and describes how to upgrade your ContactManager configuration from a previous major version. The intended readers are system administrators and implementation engineers who must merge base application changes into existing ContactManager application extensions and integrations. The <i>Configuration Upgrade Guide</i> is published with the Upgrade Tools and is available from the Guidewire Community.
<i>New and Changed Guide</i>	Describes new features and changes from prior PolicyCenter versions. Intended readers are business users and system administrators who want an overview of new features and changes to features. Consult the “Release Notes Archive” part of this document for changes in prior maintenance releases.
<i>Installation Guide</i>	Describes how to install PolicyCenter. The intended readers are everyone who installs the application for development or for production.
<i>System Administration Guide</i>	Describes how to manage a PolicyCenter system. The intended readers are system administrators responsible for managing security, backups, logging, importing user data, or application monitoring.
<i>Configuration Guide</i>	The primary reference for configuring initial implementation, data model extensions, and user interface (PCF) files for PolicyCenter. The intended readers are all IT staff and configuration engineers.
<i>PCF Reference Guide</i>	Describes ContactManager PCF widgets and attributes. The intended readers are configuration engineers.
<i>Data Dictionary</i>	Describes the ContactManager data model, including configuration extensions. The dictionary can be generated at any time to reflect the current ContactManager configuration. The intended readers are configuration engineers.
<i>Security Dictionary</i>	Describes all security permissions, roles, and the relationships among them. The dictionary can be generated at any time to reflect the current ContactManager configuration. The intended readers are configuration engineers.
<i>Globalization Guide</i>	Describes how to configure ContactManager for a global environment. Covers globalization topics such as global regions, languages, date and number formats, names, currencies, addresses, and phone numbers. The intended readers are configuration engineers who localize ContactManager.
<i>Rules Guide</i>	Describes business rule methodology and the rule sets in Guidewire Studio for PolicyCenter. The intended readers are business analysts who define business processes, as well as programmers who write business rules in Gosu.

Document	Purpose
<i>Contact Management Guide</i>	Describes how to configure Guidewire InsuranceSuite applications to integrate with ContactManager and how to manage client and vendor contacts in a single system of record. The intended readers are ContactManager implementation engineers and ContactManager administrators.
<i>Best Practices Guide</i>	A reference of recommended design patterns for data model extensions, user interface, business rules, and Gosu programming. The intended readers are configuration engineers.
<i>Integration Guide</i>	Describes the integration architecture, concepts, and procedures for integrating PolicyCenter with external systems and extending application behavior with custom programming code. The intended readers are system architects and the integration programmers who write web services code or plugin code in Gosu or Java.
<i>Java API Reference</i>	Javadoc-style reference of ContactManager Java plugin interfaces, entity fields, and other utility classes. The intended readers are system architects and integration programmers.
<i>Gosu Reference Guide</i>	Describes the Gosu programming language. The intended readers are anyone who uses the Gosu language, including for rules and PCF configuration.
<i>Gosu API Reference</i>	Javadoc-style reference of ContactManager Gosu classes and properties. The reference can be generated at any time to reflect the current ContactManager configuration. The intended readers are configuration engineers, system architects, and integration programmers.
<i>Glossary</i>	Defines industry terminology and technical terms in Guidewire documentation. The intended readers are everyone who works with Guidewire applications.
<i>Product Model Guide</i>	Describes the PolicyCenter product model. The intended readers are business analysts and implementation engineers who use PolicyCenter or Product Designer. To customize the product model, see the <i>Product Designer Guide</i> .
<i>Product Designer Guide</i>	Describes how to use Product Designer to configure lines of business. The intended readers are business analysts and implementation engineers who customize the product model and design new lines of business.

Conventions in this document

Text style	Meaning	Examples
<i>italic</i>	Indicates a term that is being defined, added emphasis, and book titles. In monospace text, italics indicate a variable to be replaced.	<p>A <i>destination</i> sends messages to an external system.</p> <p>Navigate to the ContactManager installation directory by running the following command:</p> <pre>cd installDir</pre>
bold	Highlights important sections of code in examples.	<pre>for (i=0, i<someArray.length(), i++) { newArray[i] = someArray[i].getName() }</pre>
narrow bold	The name of a user interface element, such as a button name, a menu item name, or a tab name.	Click Submit .
monospace	Code examples, computer output, class and method names, URLs, parameter names, string literals, and other objects that might appear in programming code.	The getName method of the IDoStuff API returns the name of the object.
<i>monospace italic</i>	Variable placeholder text within code examples, command examples, file paths, and URLs.	<p>Run the startServer <i>server_name</i> command.</p> <p>Navigate to http://<i>server_name</i>/index.html.</p>

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

New and changed features in ContactManager

This topic presents information on what is new and changed in ContactManager.

This topic is intended for system administrators and business users who want to upgrade to ContactManager 8.0 and need an understanding of the new features and changes in the releases. The assumption is that you are familiar with using previous releases of ContactManager.

New and changed features in ContactManager 10.0.0

There are no new features in ContactManager 10.0.0. There are changes that affect integration and compatibility with core applications.

See also

- For major issues found in previous releases and fixed in ContactManager 10.0.0 and for known issues in ContactManager 10.0.0, see the Guidewire ContactManager 10.0.0 release notes.

Changes in ContactManager 10.0.0

The changes in ContactManager 10.0.0 are to the locations of web services, web services, helper classes, and WSDL files.

ContactManager 10.0 compatibility with 9.0.x core applications

To use ContactManager 10.0 with 9.0.x versions of core applications, you use the ab900 versions of the classes used to integrate with core applications. In the 9.0.x core applications, you use the ab900 versions of the classes used to integrate with ContactManager.

See also

- “Changes in ContactManager integration files” on page 25
- “Core application web service integration with ContactManager 10.0” on page 26
- “ContactManager 10.0 web services, helper classes, WSDL files” on page 25
- “ClaimCenter classes for ContactManager integration” on page 26
- “PolicyCenter classes for ContactManager integration” on page 27
- “BillingCenter classes for ContactManager integration” on page 28

ContactManager 10.0 web services, helper classes, WSDL files

The following web services, APIs, helper classes, and WSDL files are now versioned in xx1000 folders for use with 10.0 core applications:

- gsrc/gw/contactmapper/ab1000/ContactMapper
- gsrc/gw/plugin/billing/cc1000/BCBillingSystemPlugin
- gsrc/gw/plugin/claim/cc1000/CCClaimSystemPlugin
- gsrc/gw/plugin/policy/cc1000/PCPolicySystemPlugin
- gsrc/gw/webservice/ab/ab1000/MaintenanceToolsAPI
- gsrc/gw/webservice/ab/ab1000/MessagingToolsAPI
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPIAddressSearch
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPIDocumentInfo
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPIDocumentSearchCriteria
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPIDocumentSearchResultContainer
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPIFindDuplicatesResult
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPIFindDuplicatesResultContainer
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPIPendingContactChange
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPIProximitySearchParameters
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPIRelatedContact
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPISearchCriteria
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPISearchResult
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPISearchResultContainer
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPISearchSortColumn
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPISearchSpec
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPISpecialistService
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPISubtypeFilter
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPITagMatcher
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPIUtil
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ABContactAPIValidateCreateContactResult
- gsrc/gw/webservice/ab/ab1000/abcontactapi/AddressInfo
- gsrc/gw/webservice/ab/ab1000/abcontactapi/ExceptionHandler
- gsrc/gw/webservice/ab/ab1000/abcontactapi/RelatedContactInfoContainer
- gsrc/gw/webservice/ab/ab1000/abvendorevaluationapi/ABVendorEvaluationAPI
- gsrc/gw/webservice/ab/ab1000/abvendorevaluationapi/ABVendorEvaluationAPIReviewSummary
- gsrc/gw/webservice/contactapi/ab1000/ABClientAPI
- gsrc/gw/webservice/contactapi/ab1000/ABClientAPIAddressBookUIDContainer
- gsrc/gw/webservice/contactapi/ab1000/ABClientAPIAddressBookUIDTuple
- gsrc/gw/webservice/contactapi/ab1000/ABClientAPIPendingChangeContext
- gsrc/wsi/local/gw/webservice/ab/ab1000/MaintenanceToolsAPI.wsdl
- gsrc/wsi/local/gw/webservice/ab/ab1000/MessagingToolsAPI.wsdl
- gsrc/wsi/local/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI.wsdl
- gsrc/wsi/local/gw/webservice/ab/ab1000/abvendorevaluationapi/ABVendorEvaluationAPI.wsdl
- gsrc/wsi/remote/gw/webservice/bc/bc1000.wsc
- gsrc/wsi/remote/gw/webservice/cc/cc1000.wsc
- gsrc/wsi/remote/gw/webservice/pc/pc1000.wsc

Changes in ContactManager integration files

The following table shows the files to use to integrate with compatible core applications.

For a list showing the new xx1000 location of web services and related files, see “ContactManager 10.0 web services, helper classes, WSDL files” on page 25.

ContactManager 9.0.0	ContactManager 10.0.0
BCBillingSystemPlugin (bc900 version) Plugin class that extends ClientSystemPlugin900. Package name – gw.plugin.billing.bc900 When BillingCenter 9.0 is installed with ContactManager 10.0, register this plugin implementation.	BCBillingSystemPlugin (bc1000 version) Plugin class that extends ClientSystemPlugin1000. Package name – gw.plugin.billing.bc1000 When BillingCenter 10.0 is installed with ContactManager 10.0, register this plugin implementation.
CCClaimSystemPlugin (cc900 version) Plugin class to register when ClaimCenter 9.0 is installed. Extends ClientSystemPlugin900. Package name – gw.plugin.claim.cc900 When ClaimCenter 9.0 is installed with ContactManager 10.0, register this plugin implementation.	CCClaimSystemPlugin (cc1000 version) Plugin class to register when ClaimCenter 10.0 is installed. Extends ClientSystemPlugin1000. Package name – gw.plugin.claim.cc1000 When ClaimCenter 10.0 is installed with ContactManager 10.0, register this plugin implementation.
PCPolicySystemPlugin (pc900 version) Plugin class that extends ClientSystemPlugin900. Package name – gw.plugin.policy.pc900 When PolicyCenter 9.0 is installed with ContactManager 10.0, register this plugin implementation.	PCPolicySystemPlugin (pc1000 version) Plugin class that extends ClientSystemPlugin1000. Package name – gw.plugin.policy.pc1000 When PolicyCenter 10.0 is installed with ContactManager 10.0, register this plugin implementation.

Changes to core application integration with ContactManager 10.0.0

The core applications use versioned classes, plugins, and web services to integrate with ContactManager. There are changes in core application files that you use to integrate with ContactManager.

Core application web service integration with ContactManager 10.0

ClaimCenter

ClaimCenter uses `wsi.remote.gw.webservice.ab.ab1000.wsc` to access the following ContactManager web services:

- `${ab}/ws/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI?wsdl`.
- `${ab}/ws/gw/webservice/ab/ab1000/abvendorevaluationapi/ABVendorEvaluationAPI?wsdl`.

See “Integrate ClaimCenter with ContactManager” on page 61.

PolicyCenter

PolicyCenter uses `wsi.remote.gw.webservice.ab.ab1000.wsc` to access the ContactManager web service `${ab}/ws/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI?wsdl`.

See “Integrate PolicyCenter with ContactManager” on page 67.

BillingCenter

BillingCenter uses `wsi.remote.gw.webservice.ab.ab1000.wsc` to access the ContactManager web service `${ab}/ws/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI?wsdl`.

See “Integrate BillingCenter with ContactManager” on page 73.

ClaimCenter classes for ContactManager integration

For Integration with ContactManager 9.0	For Integration with ContactManager 10.0
ABContactSystemPlugin (ab900 version) Plugin class that implements ContactSystemPlugin.java	ABContactSystemPlugin (ab1000 version) Plugin class that implements ContactSystemPlugin.java

For Integration with ContactManager 9.0	For Integration with ContactManager 10.0
Package name – gw.plugin.contact.ab900	Package name – gw.plugin.contact.ab1000
	See also <ul style="list-style-type: none"> “ABContactAPI web service” on page 264
ContactMapper.gs Package name – gw.contactmapper.ab900	ContactMapper.gs Package name – gw.contactmapper.ab1000 See “ContactMapper class” on page 275.
CCNameMapper.gs Package name – gw.contactmapper.ab900	CCNameMapper.gs Package name – gw.contactmapper.ab1000 See “Core application mapping” on page 151.
ContactSearchMapper (ab900 version) Converts the ContactSearchCriteria entity into the ABContactAPISearchCriteria that is sent to ContactManager for search operations. Package name – gw.plugin.contact.ab900	ContactSearchMapper (ab1000 version) Converts the ContactSearchCriteria entity into the ABContactAPISearchCriteria that is sent to ContactManager for search operations. Package name – gw.plugin.contact.ab1000 See “ClaimCenter support for contact searches” on page 94.
ContactSearchResultMapper (ab900 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab900	ContactSearchResultMapper (ab1000 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab1000 See “ClaimCenter support for contact searches” on page 94.
ContactAPI.gs Package name – gw.webservice.cc.cc900.contact	ContactAPI.gs Package name – gw.webservice.cc.cc1000.contact See “ABClientAPI interface” on page 272.

PolicyCenter classes for ContactManager integration

For Integration with ContactManager 9.0	For Integration with ContactManager 10.0
ABContactSystemPlugin (ab900 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab900	ABContactSystemPlugin (ab1000 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab1000 See “Integrating ContactManager with PolicyCenter in Quick-Start” on page 66.
ContactMapper.gs Package name – gw.contactmapper.ab900	ContactMapper.gs Package name – gw.contactmapper.ab1000 See “ContactMapper class” on page 275.
PCNameMapper.gs Package name – gw.contactmapper.ab900	PCNameMapper.gs Package name – gw.contactmapper.ab1000 See “Core application mapping” on page 151.
ABContactAPISearchCriteriaEnhancement (ab900 version) Converts the ContactSearchCriteria entity into the ABContactAPISearchCriteria that is sent to ContactManager for search operations. Package name – gw.plugin.contact.ab900	ABContactAPISearchCriteriaEnhancement (ab1000 version) Converts the ContactSearchCriteria entity into the ABContactAPISearchCriteria that is sent to ContactManager for search operations. Package name – gw.plugin.contact.ab1000
ABContactAPISearchResultEnhancement (ab900 version)	ABContactAPISearchResultEnhancement (ab1000 version)

For Integration with ContactManager 9.0	For Integration with ContactManager 10.0
Package name – gw.plugin.contact.ab900	Package name – gw.plugin.contact.ab1000
ContactAPI.gs Package name – gw.webservice.pc.pc900.contact	ContactAPI.gs Package name – gw.webservice.pc.pc1000.contact See “ABClientAPI interface” on page 272.

BillingCenter classes for ContactManager integration

For Integration with ContactManager 9.0	For Integration with ContactManager 10.0
ABContactSystemPlugin (ab900 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab900	ABContactSystemPlugin (ab1000 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab1000 See “Integrating ContactManager with BillingCenter in Quick-Start” on page 72.
ContactMapper.gs Package name – gw.contactmapper.ab900	ContactMapper.gs Package name – gw.contactmapper.ab1000 See “ContactMapper class” on page 275.
BCNameMapper.gs Package name – gw.contactmapper.ab900 See “Core application mapping” on page 151.	BCNameMapper.gs Package name – gw.contactmapper.ab1000 See “Core application mapping” on page 151.
ABContactAPISearchCriteriaInfoEnhancement (ab900 version) Converts the ContactSearchCriteria entity into the ABContactAPISearchCriteria object that is sent to ContactManager for search operations. Package name – gw.plugin.contact.ab900	ABContactAPISearchCriteriaInfoEnhancement (ab1000 version) Converts the ContactSearchCriteria entity into the ABContactAPISearchCriteria object that is sent to ContactManager for search operations. Package name – gw.plugin.contact.ab1000
ContactResultFromSearch (ab900 version) Package name – gw.plugin.contact.ab900	ContactResultFromSearch (ab1000 version) Package name – gw.plugin.contact.ab1000
ContactAPI.gs Package name – gw.webservice.bc.bc900.contact	ContactAPI.gs Package name – gw.webservice.bc.bc1000.contact See “ABClientAPI interface” on page 272.

New and changed features in ContactManager 9.0.5

There are no new features or major changes in ContactManager 9.0.5.

- For resolved issues and known issues in ContactManager 9.0.5, see the Guidewire ContactManager 9.0.5 Release Notes.

New and changed features in ContactManager 9.0.4

There is one new feature in ContactManager 9.0.4, Personal Data Destruction.

For resolved issues and known issues in ContactManager 9.0.4, see the ContactManager 9.0.4 release notes.

Personal Data Destruction

ContactManager supports destruction of some kinds of data. Destruction can mean either purging the data completely from the database or it can mean obfuscating data, making the original contents permanently unreadable. Guidewire recognizes the need for insurers to be able to destroy personal information both on an on-demand basis or on a time-based basis. Destruction can be mandated by regulation or business practices, within the requirements of regulation, codes of conduct, or other business practices.

See also

- “Configuring personal data destruction in ContactManager” on page 217

New and changed features in ContactManager 9.0.3

There are no new features or major changes in ContactManager 9.0.3.

- For resolved issues and known issues in ContactManager 9.0.3, see “Guidewire ContactManager 9.0.3 release notes” on page 317.

New and changed features in ContactManager 9.0.2

There are no new features or major changes in ContactManager 9.0.2.

- For resolved issues and known issues in ContactManager 9.0.2, see the “Guidewire ContactManager 9.0.2 release notes” on page 321.

New and changed features in ContactManager 9.0.1

There are no new features or major changes in ContactManager 9.0.1.

- For resolved issues and known issues in ContactManager 9.0.1, see “Guidewire ContactManager 9.0.1 release notes” on page 327.

New and changed features in ContactManager 9.0.0

There are new and changed features in ContactManager 9.0.0 that affect integration and compatibility with core applications.

See also

- For major issues found in previous releases and fixed in ContactManager 9.0.0 and for known issues in ContactManager 9.0.0, see “Guidewire ContactManager 9.0.0 release notes” on page 334.

New features in ContactManager 9.0.0

There are no new features in ContactManager for this release.

Changes in ContactManager 9.0.0

There are major changes in ContactManager 9.0.0. The changes affect use of Apache Ant, location of web services, functionality of files and packages, and changes to the ABContact data model.

ContactManager 9.0 compatibility with 8.0.x core applications

To use ContactManager 9.0 with 8.0.x versions of core applications, you use the ab800 or ab801 versions of the classes used to integrate with core applications. In the 8.0.x core applications, you use the ab800 or ab801 versions of the classes used to integrate with ContactManager.

See also

- “Changes in ContactManager file and package functionality” on page 31
- “ClaimCenter classes for ContactManager integration” on page 33
- “PolicyCenter classes for ContactManager integration” on page 33
- “BillingCenter classes for ContactManager integration” on page 34

Apache Ant installation requirement removed

Apache Ant is no longer required for general configuration and administration tasks.

The only reason to install Ant is if you are developing OSGi plugins by using the OSGi Editor in IntelliJ IDEA.

See also

- For information on using IntelliJ IDEA with the OSGi editor to deploy an OSGi plugin, see the *Integration Guide*.

Changes in location of ab800 and ab801 web services, helper classes, and WSDL files

The following web services, APIs, helper classes, and WSDL files are now versioned in xx900 folders for use with 9.0 applications:

- gw/contactmapper/ab900/ContactMapper
- gw/plugin/billing/cc900/BCBillingSystemPlugin
- gw/plugin/claim/cc900/CCClaimSystemPlugin
- gw/plugin/policy/cc900/PCPolicySystemPlugin
- gw/webservice/ab/ab900/MaintenanceToolsAPI.gs
- gw/webservice/ab/ab900/MessagingToolsAPI.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPI.gs

The ABContactAPI web service has a new method supporting search for vendor documents that ClaimCenter uses in ABContactSystemPlugin:

- retrieveDocumentsforContact – Retrieves read-only information about documents associated with vendor contacts.
- gw/webservice/ab/ab900/abcontactapi/ABContactAPIAddressSearch.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPIFindDuplicatesResult.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPIFindDuplicatesResultContainer.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPIPendingContactChange.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPIProximitySearchParameters.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPIRelatedContact.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPISearchCriteria.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPISearchResult.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPISearchResultContainer.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPISearchSortColumn.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPISearchSpec.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPISpecialistService.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPISubtypeFilter.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPITagMatcher.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPIUtil.gs
- gw/webservice/ab/ab900/abcontactapi/ABContactAPIValidateCreateContactResult.gs
- gw/webservice/ab/ab900/abcontactapi/AddressBookUIDContainer.gs
- gw/webservice/ab/ab900/abcontactapi/AddressBookUIDTuple.gs
- gw/webservice/ab/ab900/abcontactapi/AddressInfo.gs

Note: The CEDEX field of the AddressInfo class is now a Boolean value. In previous releases, it was a String.

- gw/webService/ab/ab900/abcontactapi/ExceptionHandler.gs
- gw/webService/ab/ab900/abcontactapi/RelatedContactInfoContainer.gs
- gw/webService/ab/ab900/abvendorevaluationapi/ABVendorEvaluationAPI.gs
- gw/webService/ab/ab900/abvendorevaluationapi/ABVendorEvaluationAPIReviewSummary.gs
- gw/webService/contactapi/ab900/ABClientAPI
- gw/webService/contactapi/ab900/ABClientAPIPendingChangeContext
- gsrc/wsi/local/gw/webService/ab/ab900/MaintenanceToolsAPI.wsdl
- gsrc/wsi/local/gw/webService/ab/ab900/MessagingToolsAPI.wsdl
- gsrc/wsi/local/gw/webService/ab/ab900/abcontactapi/ABContactAPI.wsdl
- gsrc/wsi/local/gw/webService/ab/ab900/abvendorevaluationapi/ABVendorEvaluationAPI.wsdl
- gsrc/wsi/remote/gw/webService/bc/bc900.wsc
- gsrc/wsi/remote/gw/webService/cc/cc900.wsc
- gsrc/wsi/remote/gw/webService/pc/pc900.wsc

See also

- “ABContactAPI web service” on page 264

Changes in ContactManager file and package functionality

The following table shows files and classes that have had changes in functionality for 9.0. See the previous topic for a list showing the new ab900 location of web services and related files.

ContactManager 8.0.0	ContactManager 9.0.0
ABContactAPI.gs Package name – gw.webService.ab.ab800.abcontactapi	ABContactAPI.gs This web service has been updated to support documents associated with contacts. Package name – gw.webService.ab.ab900.abcontactapi
See also: <ul style="list-style-type: none"> • “ABContactAPI web service” on page 264 	
ABVendorEvaluationAPI.gs Package name – gw.webService.ab.ab800.abvendorevaluationapi See “ABVendorEvaluationAPI web wervice” on page 268	ABVendorEvaluationAPI.gs A WS-I compliant web service. ClaimCenter uses this service to communicate information on vendor service provider reviews with ContactManager. The only changes in this version are that logger messages are all warnings. Package name – gw.webService.ab.ab900.abvendorevaluationapi See “ABVendorEvaluationAPI web wervice” on page 268
BCBillingSystemPlugin (bc800 version) Plugin class that extends ClientSystemPlugin800.gs. Package name – gw.plugin.billing.bc800 When BillingCenter 9.0 is installed with ContactManager 8.0, register this plugin implementation.	BCBillingSystemPlugin (bc900 version) Plugin class that extends ClientSystemPlugin900.gs. Package name – gw.plugin.billing.bc900. When BillingCenter 9.0 is installed with ContactManager 9.0, register this plugin implementation.
CCClaimSystemPlugin (cc800 version) Plugin class to register when ClaimCenter 8.0 is installed. Extends ClientSystemPlugin800.gs. Package name – gw.plugin.claim.cc800. When ClaimCenter 9.0 is installed with ContactManager 8.0, register this plugin implementation.	CCClaimSystemPlugin (cc900 version) Plugin class to register when ClaimCenter 9.0 is installed. Extends ClientSystemPlugin900.gs. Package name – gw.plugin.claim.cc900 When ClaimCenter 9.0 is installed with ContactManager 9.0, register this plugin implementation.

ContactManager 8.0.0	ContactManager 9.0.0
PCPolicySystemPlugin (pc800 version) Plugin class that extends ClientSystemPlugin800.gs. Package name – gw.plugin.policy.pc800. When PolicyCenter 9.0 is installed with ContactManager 8.0, register this plugin implementation.	PCPolicySystemPlugin (pc900 version) Plugin class that extends ClientSystemPlugin900.gs. Package name – gw.plugin.policy.pc900. When PolicyCenter 9.0 is installed with ContactManager 9.0, register this plugin implementation.
ABClientAPI.gs (ab800 version) Package name – gw.webservice.contactapi.ab800	ABClientAPI.gs (ab900 version) This interface has been updated to use the ab900 version of ABClientAPIPendingChangeContext. Package name – gw.webservice.contactapi.ab900 See “ABClientAPI interface” on page 272.

Changes in ABContact data model

To support linking documents to vendor contacts, the ABContact data model now links to an array of ABContactDocumentLink objects.

Changes to core application integration with ContactManager 9.0.0

The core applications use versioned classes, plugins, and web services to integrate with ContactManager. There are changes in core application files that you use to integrate with ContactManager.

Core application web service integration with ContactManager

- ClaimCenter uses `wsi.remote.gw.webservice.ab.ab900.wsc` to access the ContactManager web service `{ab}/ws/gw/webservice/ab/ab900/abcontactapi/ABContactAPI?wsdl`. See “Integrate ClaimCenter with ContactManager” on page 61.
- PolicyCenter uses `wsi.remote.gw.webservice.ab.ab900.wsc` to access the ContactManager web service `{ab}/ws/gw/webservice/ab/ab900/abcontactapi/ABContactAPI?wsdl`. See “Integrate PolicyCenter with ContactManager” on page 67.
- BillingCenter uses `wsi.remote.gw.webservice.ab.ab900.wsc` to access the ContactManager web service `{ab}/ws/gw/webservice/ab/ab900/abcontactapi/ABContactAPI?wsdl`. See “Integrate BillingCenter with ContactManager” on page 73.

Changes to AddressBookUID and LinkID

A new data type, `externalid`, has been introduced to handle the size of the AddressBookUID and LinkID columns in the database in a way that is consistent across application integrations. The AddressBookUID column has been added to the AddressBookLinkable delegate and has been given this type. Additionally, in ContactManager, the LinkID column on the AddressBookConvertible delegate has had its type changed to this new data type.

It is no longer necessary to add AddressBookUID manually to all entities that implement AddressBookLinkable. If you have added AddressBookUID columns to custom entities that implement AddressBookLinkable, those columns must be removed from those entities.

Entities that use the AddressBookLinkable delegate and declare their own AddressBookUID columns will have that column removed during the upgrade process.

Note: If the declaration of the AddressBookLinkable delegate and the AddressBookUID column are done in different metadata files, the upgrade tool will not be able to detect it. In this case, the AddressBookUID column will need to be removed from the entity metadata file manually.

See also

- For more information on changes to the AddressBookUID column, see the *Upgrade Guide*.

ClaimCenter classes for ContactManager integration

For Integration with ContactManager 8.0	For Integration with ContactManager 9.0
<p>ABContactSystemPlugin (ab800 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab800.</p>	<p>ABContactSystemPlugin (ab900 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab900. There are two new methods that support vendor contact document search:</p> <ul style="list-style-type: none"> retrieveDocumentsforContact – Retrieves read-only information from ContactManager about documents associated with vendor contacts. contactSystemSupportsDocuments – Determines if the contact management system has vendor document support enabled. Returns a boolean value.
<p>ContactMapper.gs Package name – gw.contactmapper.ab800.</p>	<p>ContactMapper.gs Package name – gw.contactmapper.ab900 See “ContactMapper class” on page 275.</p>
<p>CCNameMapper.gs Package name – gw.contactmapper.ab800</p>	<p>CCNameMapper.gs Package name – gw.contactmapper.ab900 See “Core application mapping” on page 151.</p>
<p>ContactSearchMapper (ab800 version) Package name – gw.plugin.contact.ab800.</p>	<p>ContactSearchMapper (ab900 version) Converts the ContactSearchCriteria entity into the ABContactAPISearchCriteria that is sent to ContactManager for search operations. Package name – gw.plugin.contact.ab900 See “ClaimCenter support for contact searches” on page 94.</p>
<p>ContactSearchResultMapper (ab800 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab800.</p>	<p>ContactSearchResultMapper (ab900 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab900 See “ClaimCenter support for contact searches” on page 94.</p>
<p>ContactAPI.gs Package name – gw.webservice.cc.cc800.contact See “ABClientAPI interface” on page 272.</p>	<p>ContactAPI.gs Package name – gw.webservice.cc.cc900.contact See “ABClientAPI interface” on page 272.</p>

See also

- “ABContactAPI web service” on page 264

PolicyCenter classes for ContactManager integration

For Integration with ContactManager 8.0	For Integration with ContactManager 9.0
<p>ABContactSystemPlugin (ab800 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab800.</p>	<p>ABContactSystemPlugin (ab900 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab900. See “Integrating ContactManager with PolicyCenter in QuickStart” on page 66.</p>

For Integration with ContactManager 8.0	For Integration with ContactManager 9.0
ContactMapper.gs Package name – gw.contactmapper.ab800.	ContactMapper.gs Package name – gw.contactmapper.ab900 See “ContactMapper class” on page 275.
PCNameMapper.gs Package name – gw.contactmapper.ab800	PCNameMapper.gs Package name – gw.contactmapper.ab900 See “Core application mapping” on page 151.
ABContactAPISearchCriteriaEnhancement (ab800 version) Package name – gw.plugin.contact.ab800.	ABContactAPISearchCriteriaEnhancement (ab900 version) Converts the ContactSearchCriteria entity into the ABContactAPISearchCriteria that is sent to ContactManager for search operations. Package name – gw.plugin.contact.ab900
ABContactAPISearchResultEnhancement (ab800 version) Package name – gw.plugin.contact.ab800.	ABContactAPISearchResultEnhancement (ab900 version) Package name – gw.plugin.contact.ab900
ContactAPI.gs Package name – gw.webservice.pc.pc800.contact See “ABClientAPI interface” on page 272.	ContactAPI.gs Package name – gw.webservice.pc.pc900.contact See “ABClientAPI interface” on page 272.

BillingCenter classes for ContactManager integration

For Integration with ContactManager 8.0	For Integration with ContactManager 9.0
ABContactSystemPlugin (ab800 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab800	ABContactSystemPlugin (ab900 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab900. See “Integrating ContactManager with BillingCenter in Quick-Start” on page 72.
ContactMapper.gs Package name – gw.contactmapper.ab800	ContactMapper.gs Package name – gw.contactmapper.ab900 See “ContactMapper class” on page 275.
BCNameMapper.gs Package name – gw.contactmapper.ab800	BCNameMapper.gs Package name – gw.contactmapper.ab900 See “Core application mapping” on page 151.
ABContactAPISearchCriteriaInfoEnhancement (ab800 version) Package name – gw.plugin.contact.ab800	ABContactAPISearchCriteriaInfoEnhancement (ab900 version) Converts the ContactSearchCriteria entity into the ABContactAPISearchCriteria object that is sent to ContactManager for search operations. Package name – gw.plugin.contact.ab900
ContactResultFromSearch (ab800 version) Package name – gw.plugin.contact.ab800	ContactResultFromSearch (ab900 version) Package name – gw.plugin.contact.ab900
ContactAPI.gs Package name – gw.webservice.bc.bc801.contact See “ABClientAPI interface” on page 272.	ContactAPI.gs Package name – gw.webservice.bc.bc900.contact See “ABClientAPI interface” on page 272.

Maximum length of LinkID and AddressBookUID fields has been increased

The ABContact entity field LinkID and the Contact entity field AddressBookUID can now accept more characters. The maximum length of the LinkID field and the corresponding AddressBookUID field in the core applications is now 64 characters. This increased maximum length matches the new maximum length of the PublicID field.

See also

- “Linking a contact” on page 195

New and changed features in ContactManager 8.0.7

There is one major change in ContactManager 8.0.7.

Personal Data Destruction

ContactManager supports destruction of some kinds of data. Destruction can mean either purging the data completely from the database or it can mean obfuscating data, making the original contents permanently unreadable. Guidewire recognizes the need for insurers to be able to destroy personal information both on an on-demand basis or on a time-based basis. Destruction can be mandated by regulation or business practices, within the requirements of regulation, codes of conduct, or other business practices.

See also

- “Configuring personal data destruction in ContactManager” on page 217.
- For resolved issues and known issues in ContactManager 8.0.7, see “Guidewire ContactManager 8.0.7 release notes” on page 339.

New and changed features in ContactManager 8.0.6

There is one major change in ContactManager 8.0.6.

Required changes to geocoding with Bing Maps

As described at <https://blogs.bing.com/maps/June-2016/Bing-Maps-V8-Web-Control-Released>, Microsoft retired the SOAP service Bing Maps v7 on June 30, 2017. Its replacement, Bing Maps v8, requires that existing code that uses the Bing Maps SOAP web services be implemented as REST.

For information about updating your geocoding services to REST for Bing Maps v8, there is a knowledge article available. Visit the Guidewire Community and search for knowledge article 7329, “Microsoft Ending Support of Bing Maps SOAP Services for Geocoding”.

See also

- For information on accessing the Guidewire Community, see “Support” on page 21.
- For resolved issues and known issues in ContactManager 8.0.6, see “Guidewire ContactManager 8.0.6 release notes” on page 344.

New and changed features in ContactManager 8.0.5

There are no new features or major changes in ContactManager 8.0.5.

- For resolved issues and known issues in ContactManager 8.0.5, see “Guidewire ContactManager 8.0.5 release notes” on page 352.

New and changed features in ContactManager 8.0.4

There are no new features or major changes in ContactManager 8.0.4.

New and changed features in ContactManager 8.0.3

ContactManager 8.0.3 now supports declaring an exit point separately for the web browser.

Change to configuring the ClaimCenter exit point to ContactManager

Previously, the web browser exit point from ClaimCenter to ContactManager used the URL defined in `suite-config.xml`, a configuration file that supports ContactManager suite integration. For example, the **Edit in ContactManager** button, available in some ClaimCenter contact editing screens, used this URL to open ContactManager in another web browser window.

To provide separation between web browser and application integration URL definitions, you can now define the browser exit point in `config.xml` by using the `ContactSystemURL` configuration parameter.

See also

- “Integrate ClaimCenter with ContactManager” on page 61
- For resolved issues and known issues in ContactManager 8.0.3, see “Guidewire ContactManager 8.0.3 release notes” on page 366.

New and changed features in ContactManager 8.0.2

ContactManager 8.0.2 now supports limiting the number of services specified in a search and has a change to the order of proximity based search results.

See also

- For resolved issues and known issues in ContactManager 8.0.2, see “Guidewire ContactManager 8.0.2 release notes” on page 377.

Configuring the number of services specified in a search

There is a new ContactManager configuration parameter that you can set in `config.xml`, `MaxNumberServicesInSearchQuery`. This configuration parameter, which defaults to a value of 20, limits the number of services that can be specified in a single contact search.

See also

- “Limiting the number of service elements specified in a Contact search” on page 94

Change to order of proximity based search results

Proximity search in ContactManager has been changed for webservice searches to always return the search results ordered by distance, smallest to largest. Any other type of sorting is discarded for proximity searches.

See also

- “Geocoding and proximity search for vendor contacts” on page 108

New and changed features in ContactManager 8.0.1

ContactManager 8.0.1 has new features and major changes.

See also

- For resolved issues and known issues in ContactManager 8.0.1, see “Guidewire ContactManager 8.0.1 release notes” on page 388.

Core applications can specify a unique id when creating a contact

Previous to this release, ContactManager was the only application that could create unique IDs for new contacts that it created. Now it is possible for a core application to send ContactManager a unique ID with a `createContact` request. If ContactManager detects that an external unique ID is specified, it uses that ID as the `LinkID` for the new contact. Otherwise, ContactManager creates a `LinkID` for the new contact, as it did previously.

If ContactManager detects that the unique ID specified by a core application is already in use, ContactManager throws an exception and does not create the contact. The calling application must recover from this state, either by providing a new unique ID or allowing ContactManager to create a unique ID.

PolicyCenter is the only core application that implements this functionality in its base configuration. PolicyCenter sends new contacts both to ContactManager and to BillingCenter with messages that are asynchronous. To ensure that both BillingCenter and ContactManager reference the same contact, PolicyCenter must specify the unique ID itself, rather than letting ContactManager create one.

See also

- “Creating and linking a contact” on page 195
- “ContactManager link IDs and comparison to other IDs” on page 260

Changes to contact linking and synchronizing APIs in ClaimCenter

Significant changes to contact APIs resulted from the change in contact system plugin interfaces from `IAddressBookAdapter` to `ContactSystemPlugin`. See “ClaimCenter integration file name, method, and package changes” on page 43. Included in these changes were deprecation of the methods `Contact.sync`, `Contact.linked`, and `Contact.synced`.

Instead of calling these methods on the contact entity itself, use the following methods from `gw.api.contact.ContactSystemUtil`:

- `ContactSystemUtil.INSTANCE.generateLinkStatus(aContact)`
- `ContactSystemUtil.INSTANCE.syncToContactSystem(aContact)`
- `ContactSystemUtil.INSTANCE.contactIsLinked(aContact)` – See following note about improving performance.
- `ContactSystemUtil.INSTANCE.contactIsSynced(aContact)` – See following note about improving performance.

Note

For better performance, first call `ContactSystemUtil.INSTANCE.generateLinkStatus(aContact)` and then query the returned `ContactSystemLinkStatus` object for the status you are interested in, such as `linked` or `synced`. For example:

```
var linkStatus = ContactSystemUtil.INSTANCE.generateLinkStatus(theContact)
if (linkStatus.isLinked() && linkStatus.isSynced()) {...
```

It is more expensive to call `ContactSystemUtil.INSTANCE.contactIsLinked(aContact)` and then call `ContactSystemUtil.INSTANCE.contactIsSynced(aContact)`. Calling those methods on a `Contact` results in two round trips between ClaimCenter and ContactManager.

Approving and editing a pending update or create

There is a new button on the screens for pending updates and pending creates, **Approve Then Edit**. Click this button if you want to make further changes to contact data after you approve the create or update. See “Review pending changes to contacts” on page 253.

Finding duplicate contacts for a pending create

There is a new button on the screens for pending creates, **Find Duplicates**. Click this button if you want to see if a contact in the pending create list is already in ContactManager. See “Review pending changes to contacts” on page 253.

Merging and editing potential duplicate contacts

There is a new button on the screens for merging potential duplicate contacts, **Merge Then Edit**. Click this button if you want to make further changes to contact data after you merge two contacts. See “Merge duplicate contacts” on page 251.

Changing the subtype of a Contact instance

WARNING This feature must be used with caution by experienced database and configuration professionals. It requires that ContactManager and ClaimCenter be in maintenance mode. The change of subtype directly affects the database and makes it impossible to synchronize the contact with ClaimCenter until you made the same change in both ClaimCenter and ContactManager. Additionally, the change can prevent users from editing the contact until you delete fields that are not compatible with the contact subtype.

You can change the subtype of a contact that was created with the wrong Contact subtype without having to delete and re-create the contact. For example, you created a contact with subtype **Doctor** when you intended the contact to be an **Attorney**. It can be less work, and safer, to delete the contact and re-create it with the correct subtype. However, deleting and re-creating might not be practical if the contact has history that you must preserve, such as being the payee on a check.

This feature is available through a new command-prompt utility. See “Changing the subtype of a Contact instance” on page 185.

Using an Address field to search for contacts

You can now add Address fields to ContactManager and core applications searches. See “Adding an address field to Contact search” on page 101.

Changes to ab800 web services, helper classes, and WSDL files

The following web services, APIs, and helper classes changed between ContactManager 8.0.0 and 8.0.1. The ab800 node in the original packages was changed to ab801, as shown in the following list:

- gsrc/gw/webService/ab/ab801/MaintenanceToolsAPI.gs
- gsrc/gw/webService/ab/ab801/MessagingToolsAPI.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPI.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPIAddressSearch.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPIFindDuplicatesResult.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPIFindDuplicatesResultContainer.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPIPendingContactChange.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPIProximitySearchParameters.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPIRelatedContact.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPISearchCriteria.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPISearchResult.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPISearchResultContainer.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPISearchSortColumn.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPISearchSpec.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPISpecialistService.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPISubtypeFilter.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPITagMatcher.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPIUtil.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ABContactAPIValidateCreateContactResult.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/AddressBookUIDContainer.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/AddressBookUIDTuple.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/AddressInfo.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/ExceptionHandler.gs
- gsrc/gw/webService/ab/ab801/abcontactapi/RelatedContactInfoContainer.gs
- gsrc/wsi/local/gw/webService/ab/ab801/MaintenanceToolsAPI.wsdl
- gsrc/wsi/local/gw/webService/ab/ab801/MessagingToolsAPI.wsdl
- gsrc/wsi/local/gw/webService/ab/ab801/abcontactapi/ABContactAPI.wsdl

Changes to core application integration with ContactManager

The following core application web services that are used to integrate with ContactManager have new xx801 files or references:

- ClaimCenter now uses wsi.remote.gw.webService.ab.ab801.wsc to access the ContactManager web service `${ab}/ws/gw/webService/ab/ab801/abcontactapi/ABContactAPI?wsdl`. See “Integrate ClaimCenter with ContactManager” on page 61.
- PolicyCenter now uses wsi.remote.gw.webService.ab.ab801.wsc to access the ContactManager web service `${ab}/ws/gw/webService/ab/ab801/abcontactapi/ABContactAPI?wsdl`. See “Integrate PolicyCenter with ContactManager” on page 67.
- BillingCenter now uses wsi.remote.gw.webService.ab.ab801.wsc to access the ContactManager web service `${ab}/ws/gw/webService/ab/ab801/abcontactapi/ABContactAPI?wsdl`. See “Integrate BillingCenter with ContactManager” on page 73.
- BillingCenter has moved all its 8.0.1 web services to directories with a bc801 folder. For example,
 - The BillingCenter implementation of ABCClientAPI, ContactAPI.gs, is now in the package `gw.webService.bc.bc801.contact`. See “Synchronizing BillingCenter and ContactManager contacts” on page 202.
 - The ContactAPI.wsdl file is accessible in the Guidewire Studio™ for BillingCenter **Project** window in **configuration**→**gsrc** at `wsi.local.gw.webService.bc.bc801.contact`.

Therefore, the ContactManager web services collection `bc800.wsc`, which is in `ws1.remote.gw.webservice.bc`, now uses the following path to access the WSDL file for BillingCenter:

```
${bc}/ws/gw/webservice/bc/bc801/contact/ContactAPI?wsdl
```

See also

- “Integrate BillingCenter with ContactManager” on page 73
- “Configure ContactManager-to-BillingCenter authentication” on page 87

New and changed features in ContactManager 8.0.0

ContactManager 8.0.0 has new and changed features and changes to compatibility with core applications.

See also

- For known issues in ContactManager 8.0.0, see “Guidewire ContactManager 8.0.0 release notes” on page 400

New features in ContactManager 8.0.0

ContactManager 8.0.0 now supports pending creates and pending changes to contacts.

Pending creates and pending changes to contacts

As part of the changes in contact management implemented in ClaimCenter, there are new APIs and a new **Pending Changes** screen in the ContactManager **Contacts** tab.

- A pending create is created when a ClaimCenter user without permission to create a ContactManager contact creates a vendor contact.

When ContactManager receives a pending contact creation request for a vendor contact, it creates a new contact entity that it marks as Pending. If that new contact creation is disapproved, ContactManager must delete that entity. Vendor contact edits are not applied until approved.

- A pending change is created when a ClaimCenter user without contact edit permissions makes a change to a vendor contact.

Either action is pending until a contact administrator logs into ContactManager and either confirms or denies the pending change or create.

See also

- For information on using the new **Pending Changes** screen, see “Review pending changes to contacts” on page 253.
- For information on the new APIs, see:
 - “ABContactAPI web service” on page 264
 - “ABClientAPI interface” on page 272

Changes in ContactManager 8.0.0

ContactManager 8.0.0 has changes to plugins, web services, and classes used to integrate with core applications.

Changes to contact plugins, web services, and mapper classes

There are changes to the names and packages of the plugins, web services, and entity mapping classes that ContactManager and the core applications use for integration and communicating contact information.

See also

- For instructions on how to register core Guidewire plugin implementations to integrate with ContactManager, see “Integrating ContactManager with Guidewire core applications” on page 59.
- For a reference description of the API, see “ABContactAPI web service” on page 264.

ContactManager integration file and package changes

ContactManager 7.0	ContactManager 8.0.0
<p>ABContactAPI.gs The web service available to core applications to make contact related calls into ContactManager, such as create, retrieve, update, and delete contacts. Package name – gw.webservice.ab.ab700.abcontactapi</p>	<p>ABContactAPI.gs This web service has been updated to support services and pending contact changes. Package name – gw.webservice.ab.ab800.abcontactapi To find and open the class in Guidewire Studio™, use the Class Search dialog in non-project mode. Note: Press Ctrl+N twice to turn on search that includes non-project classes. See “ABContactAPI web service” on page 264.</p>
<p>ABClientAPI and ABContactAPI methods have TransactionID parameter. The following methods used a transactionID parameter to identify the method call to the web service:</p> <ul style="list-style-type: none"> • ABClientAPI.mergeContacts • ABClientAPI.removeContact • ABClientAPI.updateContact • ABContactAPI.createContact • ABContactAPI.removeContact • ABContactAPI.updateContact 	<p>ABClientAPI and ABContactAPI methods no longer use the TransactionID parameter. The transaction ID is now set for the SOAP header in gw.webservice.contactapi.ContactAPIUtil.setTransactionId. Instead of passing the transaction ID as part of the contact method call, it is set in a separate method. For example:</p> <pre> ContactAPIUtil.setTransactionId(ABContactAPI.Config, transactionId) ABContactAPI.updateContact(xml) </pre>
<p>IReviewSummaryAPI.gs An RPC-E web service available to core applications for creating and deleting review summaries corresponding to vendor service provider reviews in ClaimCenter. Package name – gw.webservice.ab.ab700.reviewsummary</p>	<p>ABVendorEvaluationAPI.gs A WS-I compliant web service. ClaimCenter uses this service to communicate information on vendor service provider reviews with ContactManager. Package name – gw.webservice.ab.ab800.abvendorevaluationapi To find and open the class in Guidewire Studio™, use the Class Search dialog in non-project mode. Note: Press Ctrl+N twice to turn on search that includes non-project classes. See “ABVendorEvaluationAPI web wervice” on page 268</p>
<p>ContactIntegrationXMLMapper.gs Name of the class that maps contact data as XML between ContactManager and the core applications. Package name – gw.webservice.ab.ab700.abcontactapi</p>	<p>ContactMapper This XML mapping class has been completely changed in ContactManager 8.0. Package name – gw.contactmapper.ab800 See “ContactMapper class” on page 275.</p>
	<p>ContactIntegrationXMLMapper.gs This XML mapping class supports integration with version 7.0 core applications. Package name – gw.contactmapper.ab700</p>
<p>ClientSystemPlugin</p>	<p>ClientSystemPlugin No name change. Read-only file is now visible in Guidewire Studio™ if you do a Ctrl+N search for ClientSystemPlugin.</p>

See also

- “ABClientAPI interface” on page 272
- “ABContactAPI web service” on page 264
- For information on setting transaction IDs on web services, see the *Integration Guide*.

ContactManager 7.0	ContactManager 8.0.0
Name of the plugin interface: ClientSystemPlugin.java. Package name: gw.plugin.	
ClaimSystemPlugin.xml Name of plugin registry. Navigate in Resource pane to configuration → Plugins → gw → plugin → ClientSystemPlugin	ClaimSystemPlugin.gwp Name of plugin registry. Navigate in Project window to configuration → config → Plugins → Registry
CCClaimSystemPlugin Plugin class to register when ClaimCenter 7.0 is installed. Extends AbstractClientSystemPlugin.gs. Package name – gw.plugin.claim.cc700.	CCClaimSystemPlugin (cc800 version) Plugin class to register when ClaimCenter 8.0 is installed. Extends ClientSystemPlugin800.gs. Package name – gw.plugin.claim.cc800. When ClaimCenter 8.0 is installed with ContactManager 8.0, register this plugin implementation.
	CCClaimSystemPlugin (cc700 version) Plugin class to register when ClaimCenter 7.0 is installed. Extends ClientSystemPlugin700.gs. Package name – gw.plugin.claim.cc700. When ClaimCenter 7.0 is installed with ContactManager 8.0, register this plugin implementation.
PolicySystemPlugin.xml Name of plugin registry. Navigate in Resource pane to configuration → Plugins → gw → plugin → ClientSystemPlugin .	PolicySystemPlugin.gwp Name of plugin registry. Navigate in Project window to configuration → config → Plugins → Registry .
PCPolicySystemPlugin Plugin class that extends AbstractClientSystemPlugin.gs. Package name – gw.plugin.policy.pc700	PCPolicySystemPlugin (pc800 version) Plugin class that extends ClientSystemPlugin800.gs. Package name – gw.plugin.policy.pc800 When PolicyCenter 8.0 is installed with ContactManager 8.0, register this plugin implementation.
	PCPolicySystemPlugin (pc700 version) Plugin class that extends ClientSystemPlugin700.gs. Package name – gw.plugin.policy.pc700 When PolicyCenter 7.0 is installed with ContactManager 8.0, register this plugin implementation.
BillingSystemPlugin.xml Name of plugin registry. Navigate in Resource pane to configuration → Plugins → gw → plugin → ClientSystemPlugin	BillingSystemPlugin.gwp Name of plugin registry. Navigate in Project window to configuration → config → Plugins → Registry
BCBillingSystemPlugin Plugin class that extends AbstractClientSystemPlugin.gs. Package name – gw.plugin.policy.bc700	BCBillingSystemPlugin (bc800 version) Plugin class that extends ClientSystemPlugin800.gs. Package name – gw.plugin.billing.bc800 When BillingCenter 8.0 is installed with ContactManager 8.0, register this plugin implementation.
	BCBillingSystemPlugin (bc700 version) Plugin class that extends ClientSystemPlugin700.gs. Package name – gw.plugin.billing.bc700 When BillingCenter 7.0 is installed with ContactManager 8.0, register this plugin implementation.
ABClientAPI.gs The interface implemented by core applications to provide a way for ContactManager to call into those applications.	ABClientAPI.gs (ab800 version) This interface has been updated to support pending contact changes. Package name – gw.webservice.contactapi.ab800

ContactManager 7.0	ContactManager 8.0.0
Package name – gw.webservice.ab.ab700abcontactapi	To find and open the class in Guidewire Studio™, use the Class Search dialog in non-project mode. Note Press Ctrl+N twice to turn on search that includes non-project classes. See “ABClientAPI interface” on page 272.
	ABClientAPI.gs (ab700 version) This interface supports ContactManager 7.0 and is in a new package. Package name – gw.webservice.contactapi.ab700

ClaimCenter integration file name, method, and package changes

ClaimCenter 7.0	ClaimCenter 8.0.0
IAddressBookAdapter IContactSearchAdapter Both 7.0 plugin interfaces are deprecated in ClaimCenter 8.0 and have been replaced by ContactSystemPlugin.	ContactSystemPlugin Name of both the plugin registry, ContactSystemPlugin.gwp, and the plugin interface, ContactSystemPlugin.java. See “Integrating ContactManager with ClaimCenter in Quick-Start” on page 59.
ABContactPlugin Plugin class that implements IAddressBookAdapter.java Package name – gw.plugin.addressbook.ab700	ABContactSystemPlugin (ab800 version) Plugin class that implements ContactSystemPlugin.java Package name – gw.plugin.contact.ab800 When ContactManager 8.0 is installed with ClaimCenter 8.0, register this plugin implementation.
ContactIntegrationXMLMapper.gs Name of the class that maps contact data as XML between ClaimCenter and ContactManager. Package name – gw.plugin.addressbook.ab700.mapper.	ContactMapper.gs This XML mapping class has been completely changed in ClaimCenter 8.0. Package name – gw.contactmapper.ab800 See “ContactMapper class” on page 275.
cc-to-cm-type-mapping.xml and cm-to-cc-type-mapping.xml Name of the files used to specify how to map differing entity names and typecodes between ClaimCenter and ContactManager.	CCNameMapper.gs This Gosu class replaces the two XML mapping files in ClaimCenter 8.0. Package name – gw.contactmapper.ab800 See “Core application mapping” on page 151.
ContactAPI.gs The web service that implements ABClientAPI to provide a way for ContactManager to call into ClaimCenter. Package name – gw.webservice.cc.cc700.contact	ContactAPI.gs This web service has been updated to support services and pending contact changes. Package name – gw.webservice.cc.cc800.contact See “ABClientAPI interface” on page 272.
contact-sync-config.xml When determining if a ClaimCenter contact is synchronized with its ContactManager counterpart, this file defines: <ul style="list-style-type: none"> The contact properties to exclude from the comparison The contact relationships to include in or exclude from the comparison You can specify relationships to include or exclude based on Contact subtype.	<ul style="list-style-type: none"> withAffectsSync(false) Use this method, available in the ClaimCenter ContactMapper class, to exclude fields of Contact entities and subentities from the synchronization check. RelationshipSyncConfig.gs Replaces the XML file, which was visible in PolicyCenter and BillingCenter even though those core applications did not use it. Now available only for ClaimCenter, this Gosu class provides the same functionality as the XML file through its includeRelationship and excludeRelationship methods. Package name – gw.plugin.contact.ab800 See “Synchronizing ClaimCenter contact fields” on page 207.

PolicyCenter integration file name and package changes

PolicyCenter 7.0	PolicyCenter 8.0.0
ContactSystemPlugin Name of both plugin registry, ContactSystemPlugin.xml, and plugin interface, ContactSystemPlugin.java.	ContactSystemPlugin Name of both the plugin registry, ContactSystemPlugin.gwp, and the plugin interface, ContactSystemPlugin.java See “Integrating ContactManager with PolicyCenter in QuickStart” on page 66.
ABContactSystemPlugin Plugin class that implements ContactSystemPlugin.java. Package name – gw.plugin.contact.ab700	ABContactSystemPlugin (ab800 version) Plugin class that implements ContactSystemPlugin.java. Package name – gw.plugin.contact.ab800. When ContactManager 8.0 is installed with PolicyCenter 8.0, register this plugin implementation.
	ABContactSystemPlugin (ab700 version) Plugin class that implements ContactSystemPlugin.java. Package name – gw.plugin.contact.ab700 When ContactManager 7.0 is installed with PolicyCenter 8.0, register this plugin implementation.
ContactIntegrationXMLMapper.gs Name of the class that maps contact data as XML between PolicyCenter and ContactManager. Package name – gw.plugin.contact.ab700	ContactMapper.gs This XML mapping class has been completely changed in PolicyCenter 8.0. It supports integration with ContactManager 8.0. Package name – gw.contactmapper.ab800 See “ContactMapper class” on page 275.
	ContactIntegrationXMLMapper.gs This XML mapping class supports integration with ContactManager 7.0. Package name – gw.contactmapper.ab700
pc-to-cm-type-mapping.xml and cm-to-pc-type-mapping.xml Names of the files used to specify how to map differing entity names and typecodes between PolicyCenter and ContactManager.	PCNameMapper.gs This Gosu class replaces the two XML mapping files in PolicyCenter 8.0. Package name – gw.contactmapper.ab800 See “Core application mapping” on page 151.
ContactAPI.gs The web service that implements ABClientAPI to provide a way for ContactManager to call in- to PolicyCenter. Package name – gw.webservice.pc.pc700.contact	ContactAPI.gs (ab800 version) Package name – gw.webservice.pc.pc800.contact See “ABClientAPI interface” on page 272.
	ContactAPI.gs (ab700 version) This web service supports integration with ContactManager 7.0. Package name – gw.webservice.pc.pc700.contact

BillingCenter integration file and package changes

BillingCenter 7.0	BillingCenter 8.0.0
IContactSystemPlugin Name of both plugin registry, IContactSystemPlugin.xml, and plugin interface, IContactSystemPlugin.java	ContactSystemPlugin Name of both the plugin registry, ContactSystemPlugin.gwp, and the plugin interface, ContactSystemPlugin.java See “Integrating ContactManager with BillingCenter in QuickStart” on page 72.
ContactManagerSystemPlugin	ABContactSystemPlugin (ab800 version) Plugin class that implements ContactSystemPlugin.java.

BillingCenter 7.0	BillingCenter 8.0.0
Plugin class that implements <code>IContactSystemPlugin.java</code> . Package name – <code>gw.plugin.contact.ab700</code>	Package name – <code>gw.plugin.contact.ab800</code> When ContactManager 8.0 is installed with BillingCenter 8.0, register this plugin implementation. ContactManagerSystemPlugin (ab700 version) Plugin class that implements <code>IContactSystemPlugin.java</code> . Package name – <code>gw.plugin.contact.ab700</code> When ContactManager 7.0 is installed with BillingCenter 8.0, register this plugin implementation in the registry <code>ContactSystemPlugin.gwp</code> . Note: Plugin implementation class name and plugin interface it implements are different from ab800 version.
<code>ContactIntegrationXMLMapper.gs</code> Name of the class that maps contact data as XML between BillingCenter and ContactManager. Package name – <code>gw.plugin.addressbook.impl</code>	<code>ContactMapper.gs</code> This XML mapping class has been completely changed in BillingCenter 8.0. It supports integration with ContactManager 8.0. Package name – <code>gw.contactmapper.ab800</code> See “ContactMapper class” on page 275.
	<code>ContactIntegrationXMLMapper.gs</code> This XML mapping class supports integration with ContactManager 7.0. Package name – <code>gw.contactmapper.ab700</code>
<code>bc-to-cm-type-mapping.xml</code> and <code>cm-to-bc-type-mapping.xml</code> Name of the files used to specify how to map differing entity names and typecodes between BillingCenter and ContactManager.	<code>BCNameMapper.gs</code> This Gosu class replaces the two XML mapping files in BillingCenter 8.0. Package name – <code>gw.contactmapper.ab800</code> See “Core application mapping” on page 151.
<code>ContactAPI.gs</code> The web service that implements <code>ABClientAPI</code> to provide a way for ContactManager to call in to BillingCenter. Package name – <code>gw.webservice.bc.bc700.contact</code>	<code>ContactAPI.gs</code> (ab800 version) Package name – <code>gw.webservice.bc.bc800.contact</code> See “ABClientAPI interface” on page 272.
	<code>ContactAPI.gs</code> (ab700 version) This web service supports integration with ContactManager 7.0 and has been moved to a different package. Package name – <code>gw.webservice.bc.bc700.contact</code>

Changes to search classes

There are changes to the names of search classes that ContactManager provides to core applications for searching for contacts. All the ContactManager 8.0.0 classes are in the class package `gw.webservice.ab.ab800.abcontactapi`.

ContactManager 7.0	ContactManager 8.0.0
<code>ABContactSearchCriteriaInfo.gs</code> A class that specifies the contact search criteria that the Guidewire core applications can use.	<code>ABContactAPISearchCriteria.gs</code> A class that specifies the contact search criteria that the Guidewire core applications can use. You can edit this class to add new search criteria for contacts. See “ContactManager support for core application searches” on page 94.
<code>ABContactAPISearchResult.gs</code> A class that specifies the fields included in the contact search results that ContactManager sends back to the Guidewire core application.	<code>ABContactAPISearchResult.gs</code>

ContactManager 7.0	ContactManager 8.0.0
	A class that specifies the fields included in the contact search results that ContactManager sends back to the Guidewire core application. You can edit this class to send additional search results to core applications. See “ContactManager support for core application searches” on page 94.
ABContactAPISearchSpecInfo.gs Provides the search criteria used by the web service method ABContactAPI.findDuplicates.	ABContactAPISearchSpec.gs Provides the search criteria used by the web service method ABContactAPI.findDuplicates.
ABContactSearchSortColumnInfo.gs	ABContactAPISearchSortColumn.gs
AddressSearchInfo.gs	ABContactAPIAddressSearch.gs Used by ABContactAPISearchCriteria to declare the Address and ProximitySearchCenter variables.
ProximitySearchParametersInfo.gs	ABContactAPIProximitySearchParameters.gs Used by ABContactAPISearchCriteria to declare the ProximitySearchParameters variable.

Changes to contact search functionality

There are changes to typekey criteria in contact searches that affect how searches are performed across the core applications and ContactManager.

ABContactAPI now exposes typekeys used in Contact search as strings.

In the previous release, ContactManager 7.0 exposed these typekeys as enum values that a core application could access from ContactSearchMapper as follows:

```
searchCriteriaInfo.VendorType =
    wsi.remote.gw.webservice.ab.ab700.abcontactapi.enums.VendorType.forGosuValue(
        searchCriteria.VendorType.Code)
```

In ContactManager 8.0, ABContactAPI exposes typekeys as strings, such as the VendorType typelist:

```
@WsiExposeEnumAsString(typekey.VendorType)
```

Now that these typekeys are simple strings, the core application method call is also much simpler. For example, the ClaimCenter ContactSearchMapper code for the VendorType search criterion is now:

```
searchCriteriaInfo.VendorType = searchCriteria.VendorType.Code
```

Contact entity mapping classes and XML files

The mapping class names have changed and the XML mapping files used in core applications are now Gosu classes. Specifically:

- The ContactIntegrationXMLMapper class has been refactored in ContactMapper. The new class provides:
 - Simpler code for adding foreign keys and array references
 - Ability to specify fields that determine if a contact is in sync
 - Ability to specify fields for a contact that are persisted in the core application
- The XML configuration files pc-to-cm-type-mapping.xml and cm-to-pc-type-mapping.xml have been replaced by the single Gosu class gw.contactmapper.ab800.PCNameMapper.gs.

For the tables showing the name changes, see “Changes to contact plugins, web services, and mapper classes” on page 40.

See also

- “Working with contact mapping files” on page 150
- “Core application mapping” on page 151
- “ContactMapper class” on page 275

Changes to ClaimCenter contact synchronization

Previously, you used the `IgnoreProperty` element of `contact-sync-config.xml` to exclude `Contact` fields from determining synchronization. In `ContactManager 8.0`, you no longer use this file at all.

- To exclude other fields of a contact, instead of using the `IgnoreProperty` element, you use the `ContactMapper` method `withAffectsSync`. By default, all fields that you add to `ContactMapper` are included in the fields that are checked for synchronization status unless you specifically exclude them by using `withAffectsSync`. See “Excluding properties from contact synchronization” on page 207.
- To exclude contact relationships from the set of fields `ClaimCenter` uses to determine if a contact is synchronized with `ContactManager`, use the Gosu class `RelationshipSyncConfig`. See “Including and excluding contact relationships in synchronization” on page 207.

See also

- “Synchronizing `ClaimCenter` contact fields” on page 207

ContactManager 8.0 compatibility with core applications

`ContactManager 8.0` and the Guidewire 8.0 core applications support the following compatibility scenarios:

- `ContactManager 8` and a core application with version 8.0 or later.
Note: `ClaimCenter 8.0` supports only `ContactManager 8.0` and later. The 8.0 version of `ClaimCenter` does not support `ContactManager 7.0`.
- `ContactManager 8` and `ClaimCenter 7.0.0` or later.
- `ContactManager 8` and `PolicyCenter 7.0.1` or later.
- `ContactManager 8` and `BillingCenter 7.0.0` or later.

For specific information on configuring these scenarios, see the latest 8.0.x `Contact Management Guide`.

Managing integrated contacts

Depending on your core application, you can integrate ContactManager to manage vendor contacts or client contacts or both. You typically use your Guidewire core application to work with contacts, although you can log in to ContactManager and work with them there.

Client Data Management

Client Data Management, available as an optional add-on module for Guidewire core applications, integrates and synchronizes client-related information across InsuranceSuite. Client Data Management supports a comprehensive view of the customer across core insurance processes by unifying the customer record. Some Client Data Management capabilities are supported by ContactManager.

In conjunction with Guidewire InsuranceSuite core systems, Client Data Management enables you to manage customer contact data across underwriting, policy administration, billing, and claims processes. If you license the Client Data Management add-on module, you can install ContactManager and integrate it with PolicyCenter and with other InsuranceSuite core applications. With the applications integrated, Client Data Management supports your customer and agent interactions and can enable you to maintain data integrity across different systems.

ContactManager is designed to be your system of record, the central repository for your customer data. This centralized repository stores a unique customer record and synchronizes updates across your policy, billing, and claims systems.

Various types of users, such as billing specialists, underwriters, and agents, can edit customer information in their respective applications without having to navigate to another system. You can add new clients through PolicyCenter, see current client information for policies in ClaimCenter, make updates directly in ClaimCenter, and obtain the latest client data in BillingCenter. To aid data cleanup, duplicate contact detection identifies potential duplicate contacts resulting from creating new contacts and from changing existing contacts, and enables you to merge or override these contacts.

ContactManager's comprehensive customer view, available in PolicyCenter, enables service representatives to personalize interactions to match customers' needs. The search and auto-fill features support updating of contact records, and contact history tracks which updates have been made and provides visibility into previous customer interactions. Additionally, the PolicyCenter comprehensive view enables underwriters to find a customer's policy, billing, and claim information in a single place. Underwriters no longer have to search for this information in multiple systems. And, to ensure that the proper controls are in place, user permissions dictate who can update contact information at various points in the lifecycle.

PolicyCenter role in managing client data

With the Client Data Management module, after integrating PolicyCenter with ContactManager, PolicyCenter can store client data in ContactManager. Typically, you create a new client when you open an account or create a policy,

or when you add contacts to an account or policy. Additionally, in the PolicyCenter **Contact** tab, you can view information associated with a client, such as personal details like address, date of birth, and phone numbers. On the **Account** tab, you can view contact information such as all accounts, policies, and work orders associated with the client. You can also see claims and billing information if you use ClaimCenter and BillingCenter with PolicyCenter or if you integrate this feature with another claim system or billing system.

Note: While you can create a new client when you open an account or a new policy, the client data is not immediately stored in ContactManager. Contact data is stored in ContactManager only when you bind a policy or when you associate a contact with an account that has a bound policy. Additionally, contacts associated with reinsurance treaties or programs are also stored in ContactManager.

Using the **Contact** tab, you can create new contacts, search for existing contacts, select a recently viewed contact, and change contact information. PolicyCenter additionally enables you to work with contacts in its **Account** and **Policy** screens, where you can add, remove, and update contacts in various roles.

All these contacts are stored locally with the associated account or policy. Contacts that were originally in ContactManager and contacts in accounts with at least one bound policy are linked to a central record in ContactManager. If you make changes to a linked contact, PolicyCenter saves these changes locally and to the central contact record in ContactManager.

ContactManager also notifies PolicyCenter of any changes made to linked contacts outside PolicyCenter, such as a change originating in ClaimCenter when the client makes a claim. These notifications keep a contact's records synchronized across all accounts, policies, work orders, and so on.

PolicyCenter linked addresses and side effects

PolicyCenter has a feature that enables you to share the same address with more than one contact. When you share an address across contacts in PolicyCenter, the contact addresses continue to be stored separately, but PolicyCenter links them. However, in the other Guidewire applications, contact addresses are not linked. ContactManager stores each contact address as a separate, unlinked address.

If you make a change to one of these addresses from ClaimCenter, BillingCenter, or ContactManager, there is a side effect in the base configurations. When ContactManager sends the changed contact address to PolicyCenter, PolicyCenter applies the change to the other addresses to which the original changed address is linked. PolicyCenter then sends these changed contact addresses back to ContactManager, and the result is that more than one contact gets an address change. There is no indication in ClaimCenter, BillingCenter, or ContactManager of this side effect—the linked addresses are changed automatically. You can configure the applications to behave differently.

If you change a linked address in PolicyCenter, you have the option of linking or unlinking it from other contacts' addresses. However, since this feature exists only in PolicyCenter, you do not have this option in the other Guidewire applications.

ClaimCenter role in managing client data

ClaimCenter can receive client data from PolicyCenter when ClaimCenter requests policy information during claims processing. ClaimCenter also can receive client data updates from ContactManager when the information for a client stored in ContactManager changes. For example, a client might notify a PolicyCenter customer service representative that their address has changed. The customer service representative updates the client's address, and PolicyCenter sends the change to ContactManager, which subsequently sends the change to ClaimCenter.

Additionally, a client might notify a ClaimCenter customer service representative that their information has changed, such a change of phone number or address. The ClaimCenter customer service representative can change that information for the client in a claim, and then ClaimCenter can send the change to ContactManager.

ContactManager then broadcasts the change to any Guidewire application that is integrated with ContactManager, such as PolicyCenter.

BillingCenter role in managing client data

BillingCenter can receive client data from PolicyCenter when PolicyCenter sends billing information to BillingCenter. BillingCenter also can receive client data updates from ContactManager when the information for a client stored in ContactManager changes. For example, a client might notify a PolicyCenter customer service

representative that their address has changed. The customer service representative updates the client's address, and PolicyCenter sends the change to ContactManager, which subsequently sends the change to BillingCenter.

Additionally, BillingCenter can send changes in client data to ContactManager. For example, a customer contacts a billing clerk to report a change of address.

Vendor Data Management

Vendor data management, available only in ClaimCenter, is typically a set of tasks related to adding, modifying, searching for, and deleting vendor contacts. The contacts are typically managed in a contact management system, like Guidewire ContactManager. A vendor contact is a person or company that provides services for claims.

Vendor contacts

A vendor contact is a person or company that provides services for claims. In ClaimCenter, a vendor contact can be a person like a doctor or attorney. Additionally, a vendor contact can be a company, such as a repair shop, a bank, or a hospital. Additionally, a vendor contact can be a specific place or venue for which your company frequently references the geographical location, such as a legal venue like a court house.

Using and configuring vendor data management

Vendor data management is typically a set of tasks related to adding, modifying, searching for, and deleting vendor contacts in a contact management system, like ContactManager. ClaimCenter provides various opportunities for you to create, edit, or search for contacts. For example, when entering a new claim, you can create a new vendor contact or search for a contact, such as an attorney, in the claim creation wizard. You can also create and search for contacts in other areas of the ClaimCenter application where you need to specify a contact for a claim.

You can configure screens to match company requirements, such as creating and searching for new kinds of contacts, and you can create entire new screens if needed. Or you can perform configurations like validating that an entered postal code is in the correct format.

Overview of ClaimCenter integration with ContactManager

ContactManager is a separate, stand-alone Guidewire application. You can use ContactManager as a central repository for standardizing and managing the contact data for your company. Typically, a company uses ContactManager in conjunction with ClaimCenter to manage vendor contacts that can be used in more than one claim. A claim can also have contacts that are associated only with that one claim, such as a claimant. These contacts can be tracked locally in ClaimCenter.

Centralized vendor data management

As the system of record for vendor contacts, ContactManager enables you to manage and serve vendor contact data centrally. ClaimCenter enables you to search for these centrally-maintained contacts on the **Address Book** tab and from claims. In claims, you can add contacts and edit the contact data as needed, and your changes can be saved in ContactManager.

After you integrate ContactManager and ClaimCenter, users searching for contacts have access not only to contacts stored locally in ClaimCenter, but also to contacts stored in ContactManager. For example, an adjuster working in ClaimCenter can search for an auto shop for a repair and access a list of approved service providers provided by ContactManager.

You can define different users for each Guidewire application. For example, you can have a group of ContactManager users whose primary function is to manage contact data and ensure that it is accurate. These users need not have authorization for ClaimCenter.

A synchronizing facility between the two applications ensures that when a contact changes, regardless of whether the change occurred in ContactManager or ClaimCenter, the change appears in both applications.

Which contacts to store in ContactManager

You need not configure ClaimCenter and ContactManager to store all company contacts in ContactManager. In some cases, it makes sense to centrally manage a contact, and in others it does not. ClaimCenter enables you to manage contacts both centrally in ContactManager and locally in ClaimCenter.

Typically, you store in ContactManager either contacts that you expect to use across multiple claims or contacts that require a single, definitive information source. For example, you would store service providers, vendors, such as doctors, auto repair shops, and inspectors in ContactManager. These contacts require correct tax ID data for reporting and accurate addresses for payments.

Contacts that are specific to a claim are best managed locally in ClaimCenter. These contacts appear only in specific instances and are unlikely to reappear. Additionally, they have no impact on your company's business processes, nor do they have any regulatory requirements. Two examples might be an accident witness on a claim and a bank employee who requests a policy verification.

Locally and centrally managed contacts

When you create a new contact, depending on what kind of contact it is, the contact can be stored in two ways. ClaimCenter can store the contact locally only in ClaimCenter. Or, ClaimCenter can store the contact locally in ClaimCenter and centrally in ContactManager and link the contacts. If you create a new, non-vendor contact directly on a claim, ClaimCenter can store it locally and not in ContactManager. This contact, associated only with the claim, is an *unlinked* contact.

Unlinked contacts are not associated with ContactManager. For an unlinked contact, ClaimCenter does not attempt to determine if a contact associated with one claim appears elsewhere on another claim. Thus, any unlinked contact can be a duplicate of one or more other unlinked contacts associated with different claims. When such duplicates exist, changes to one contact do not propagate to another because they are not related to one another.

Conversely, contacts stored in ContactManager, such as vendor contacts, do propagate changes when stored in ClaimCenter. In ClaimCenter, you can associate a single, centrally-managed contact with any number of claims. ClaimCenter makes a copy of each contact and stores it locally with the claim, and each copy is also linked to the ContactManager contact. Therefore, ClaimCenter can keep the data for all these copies of a contact in sync, even though there are multiple copies stored in ClaimCenter.

You can add a new contact and make the contact centrally managed and available for use with other claims. If the new contact is a vendor, it is sent automatically to ContactManager and does not require any other action in ClaimCenter to become linked.

Note: The contact might require action in ContactManager, however. Under certain circumstances, such as you not being a user with permissions to create contacts, ClaimCenter sends the contact as a pending create. The contact then requires verification from a user in ContactManager, as described later.

For example, from a contact management perspective, a claimant might be a contact that other adjusters do not reuse. Therefore, a claimant might be an unlinked contact, stored locally with the claim but not in ContactManager. Even so, you could add a linked, centrally managed claimant to a claim. For example, you could click **Add an Existing Contact** and find the contact in ContactManager database and then add that contact to the claim.

To link a contact, ClaimCenter and ContactManager define a connection between a specific record in the ClaimCenter database and another record in the ContactManager database. In effect, linking the two contacts declares that though they are in two different databases, they are the same and are to show the same information. Users of ClaimCenter can link contacts from ClaimCenter to ContactManager. ClaimCenter can also automatically link a vendor contact to ContactManager. From ContactManager, users cannot link contacts to ClaimCenter.

After a contact is linked, the contact can be updated from either application, and both applications can get the update. ContactManager notifies ClaimCenter of changes to linked contacts. A linked contact in ClaimCenter whose data has changed in ContactManager is marked as not in sync. To keep contacts in ClaimCenter current, you can copy the data for a linked contact from ContactManager. Copying causes ClaimCenter to copy the latest information from the ContactManager central repository and mark the contact as in sync.

When you update a contact in ClaimCenter that is linked to ContactManager, ClaimCenter sends the change to ContactManager. For example, you can edit a linked contact on the **Contacts** screen of a claim and then update the contact. If you have permission to edit ContactManager contacts, ClaimCenter instructs ContactManager to save the

change. If you do not have this permission, ClaimCenter instructs ContactManager to create a pending change, which then has to be reviewed by a ContactManager user.

You can unlink a contact. An unlinked contact in ClaimCenter becomes a claim-specific contact and is no longer centrally managed. However, even though the contact is no longer linked to ContactManager, a contact that you unlink continues to be in the ContactManager database.

See also

- “Linking and synchronizing contacts” on page 195

Deciding whether to use ContactManager for vendor management

In the base configuration, ClaimCenter has a registered address book plugin that is useful only for demonstration purposes. Guidewire also supplies a fully functional Gosu plugin implementation that works with ContactManager and is suitable for production. Do not go into a production environment with the demonstration plugin registered. If you decide not to use the integration with ContactManager but instead want to integrate with another, third-party contact system, you must implement a replacement plugin.

You are not required to use either ContactManager or your own contact plugin in ClaimCenter. Without a contact plugin, ClaimCenter continues to maintain its own local set of contacts in its own database, but its address book functionality changes as follows:

- The **Address Book** tab is present, but attempts to use it for searching result in an error.
- The claim **Parties Involved** screen has reduced functionality. You are unable to view the **Address Book** from the screen or copy a contact or add an existing contact from ContactManager.

The **Address Book** tab, buttons, and other related options are always present in the base application. If you want to remove them, you must edit the related PCF files.

Note: Even without having a contact management application integrated, you can use the **Search** tab to search for local contacts. However, without this integration, you cannot check for duplicate contacts.

See also

- “Client Data Management” on page 49
- “Integrating ContactManager with Guidewire core applications” on page 59
- For information on how to create your own plugin, see the *Integration Guide*

Installing ContactManager

You can install ContactManager in a development environment. The installation process is similar to the process described in the *Installation Guide*.

You can install ContactManager for development by using the Guidewire QuickStart installation setup, or you can use any supported application and database server. Use QuickStart only in a demonstration or development environment.

ContactManager must start with its own application and database server. Do not run ContactManager in the same server as a core application.

For production, you need a dedicated Java Virtual Machine (JVM) for each Guidewire application. Running a Guidewire core application and ContactManager in the same JVM can result in memory conflicts and other problems.

For production, you can install ContactManager with any supported combination of application server and database server. For more information, visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

Installing ContactManager with QuickStart for development

A QuickStart installation enables you to immediately use and test the product. The QuickStart installation topics provide only the information necessary to get ContactManager working in the QuickStart environment.

[See also](#)

Install ContactManager with QuickStart

You can install ContactManager and use the provided H2 database for development or demonstration purposes.

Before you begin

These instructions assume that you have installed your Guidewire core application as described in the *Installation Guide*.

Procedure

1. If you have not already done so, create an installation directory for ContactManager.
This guide uses `ContactManager` as the directory name.
2. Extract the ContactManager Zip file into the ContactManager installation folder.

3. If you are not using a version control system, make a read-only copy of the ContactManager installation folder.

This copy enables you to recover quickly from accidental changes that can prevent ContactManager from starting.

4. If you are reinstalling ContactManager, drop the QuickStart database created by the previous installation. At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb dropDb
```

5. At the command prompt, enter the following command:

```
gwb runServer
```

When the server has started, you see the message ***** ContactManager ready ***** in the console messages.

6. Open a browser and enter the URL <http://localhost:8280/ab>, and then log in with user name **su** and password **gw**.

Next steps

“Load sample data for ContactManager” on page 56

Load sample data for ContactManager

ContactManager provides sample data that you can load so you can test application functionality.

Before you begin


Install ContactManager as described at “Install ContactManager with QuickStart” on page 55.


About this task

Sample data can be useful for learning purposes and for completing integration examples, but it is not intended to be used in a production system. Additionally, importing sample data can make changes that you must remove before production.

Procedure

1. Start ContactManager and log in with user name **su** and password **gw**.
 - a. At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb runServer
```
 - b. Open a browser and enter the URL <http://localhost:8280/ab>.
 - c. Log in with user name **su** and password **gw**.
2. Press **Alt+Shift+T**.
3. Click the **Internal Tools** tab.
4. Click **AB Sample Data**, and then click the **Load Sample Data** button.
5. Wait to see the completion messages above the button confirming that the sample data was imported.
6. To verify that the data loaded correctly:
 - a. On the Options menu , click **Return to ContactManager**.
 - b. Click the **Administration** tab.
 - c. In the Sidebar on the left, you see **Default Root Group**. If necessary, click **Default Root Group** to show its contents, and then click **Enigma Fire and Casualty** to see the list of users that was just loaded as sample data.

7. To log in to ContactManager as a user with permission to add, edit, and delete contacts:
 - a. On the Options menu , click **Log Out**.
 - b. Log in with user name `aaggregate` and password `gw`.

Integrating ContactManager with Guidewire core applications

You can integrate ContactManager with the Guidewire core applications ClaimCenter, PolicyCenter, and BillingCenter. The Guidewire core applications communicate with ContactManager through SOAP APIs. The applications work with ContactManager through its published web service `ABContactAPI`.

Each Guidewire core application defines a plugin interface for invoking the integration layer to communicate with an external contact management system. In each application, there is a plugin class implementation of this interface—a class that implements this interface—that communicates with ContactManager. The plugin interface and the plugin class implementation are different for each Guidewire core application.

You configure the integration between ContactManager and the Guidewire core applications in Guidewire Studio™.

See also

- “Overview of ContactManager plugins” on page 283

Integrating ContactManager using QuickStart

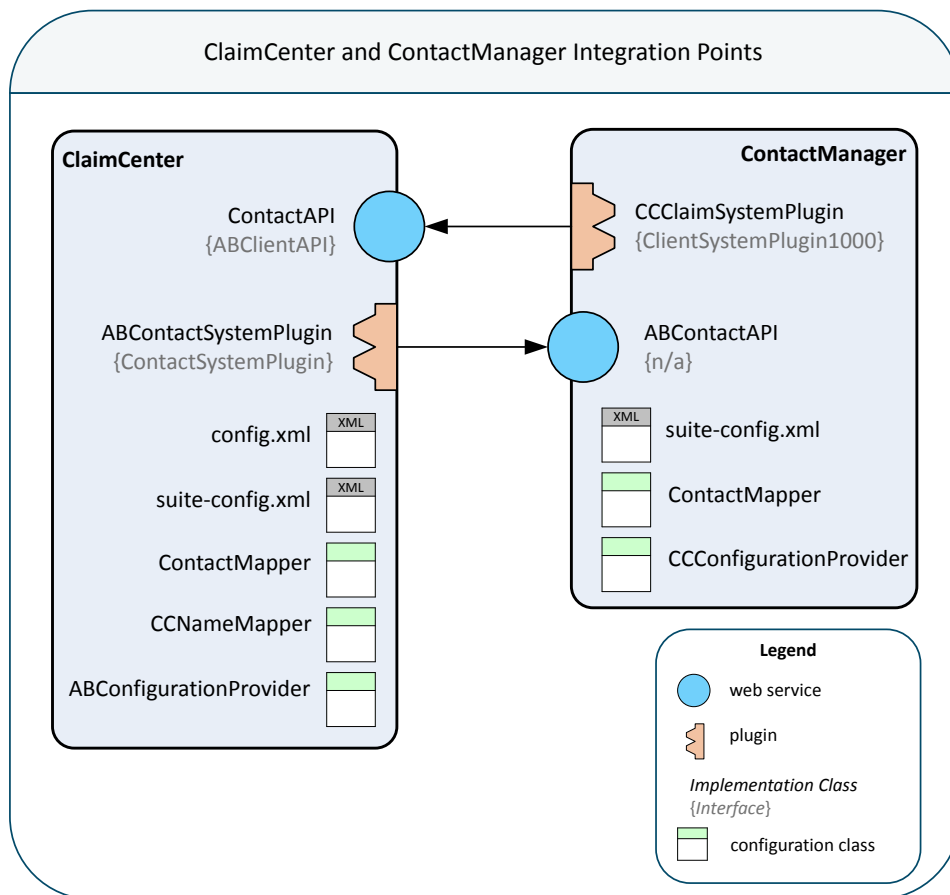
A QuickStart installation enables you to quickly use and test the product. After installing ContactManager, you integrate it with Guidewire core applications in a QuickStart environment.

See also

- “Installing ContactManager with QuickStart for development” on page 55
- For details on the `suite-config.xml` file, see the *Integration Guide*.

Integrating ContactManager with ClaimCenter in QuickStart

The following figure shows the primary integration points used to integrate ClaimCenter and ContactManager.



In general, to get the basic integration working, you edit the `suite-config.xml` files in ClaimCenter and ContactManager and the `config.xml` file in ClaimCenter. You also register the `CCClaimSystemPlugin` plugin implementation in the ContactManager plugin registry `ClaimSystemPlugin.gwp`. Additionally, you register `ABContactSystemPlugin` in the ClaimCenter registry `ContactSystemPlugin.gwp`. All these steps are described in the topics that follow.

If you extend the contact model, you must edit the `ContactMapper` classes in both ContactManager and ClaimCenter, and possibly the `CCNameMapper` class.

IMPORTANT Guidewire recommends that you change the authentication user and password. To do so, you add authentication users to ContactManager and ClaimCenter and edit the `ABConfigurationProvider` and `CCConfigurationProvider` classes.

Integrating ContactManager with ClaimCenter is a multi-step process:

1. “Integrate ClaimCenter with ContactManager” on page 61
2. “Integrate ContactManager with ClaimCenter” on page 62
3. “Test the integration of ClaimCenter and ContactManager” on page 64
4. “Troubleshooting the ClaimCenter connection with ContactManager” on page 65

See also

- “Installing ContactManager with QuickStart for development” on page 55
- “Configure ClaimCenter-to-ContactManager authentication” on page 79
- “Configure ContactManager-to-ClaimCenter authentication” on page 83

Integrate ClaimCenter with ContactManager

This step is the first in the multi-step process of integrating ClaimCenter and ContactManager.

About this task

For an overview of the integration process, see “Integrating ContactManager with ClaimCenter in QuickStart” on page 59

Procedure

1. At a command prompt, navigate to the ClaimCenter installation folder, and then start Guidewire Studio™ for ClaimCenter by entering the following command:

```
gwb studio
```

2. In Guidewire Studio for ClaimCenter, open the **Project** window.
3. Press Ctrl+Shift+N and enter `suite-config.xml` to find this file, and then double-click the search result to open the file in the editor.

This file defines the URLs of the applications in the Guidewire InsuranceSuite. If the entries in this file are commented out, you must remove the comments for the applications you have installed. In the default configuration, the Guidewire applications have the following settings in this file:

```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <!--<product name="cc" url="http://localhost:8080/cc"/>-->
  <!--<product name="pc" url="http://localhost:8180/pc"/>-->
  <!--<product name="ab" url="http://localhost:8280/ab"/>-->
  <!--<product name="bc" url="http://localhost:8580/bc"/>-->
</suite-config>
```

4. Remove the comments around the ContactManager (ab) entry, as follows:



```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <!--<product name="cc" url="http://localhost:8080/cc"/>-->
  <!--<product name="pc" url="http://localhost:8180/pc"/>-->
  <product name="ab" url="http://localhost:8280/ab"/>
  <!--<product name="bc" url="http://localhost:8580/bc"/>-->
</suite-config>
```

5. Press Ctrl+Shift+N and enter `config.xml` to find this file, and then double-click the search result to open the file in the editor.
6. Add the ContactManager URL to the `ContactSystemURL` configuration parameter.

For example:

```
<param name="ContactSystemURL" value="http://localhost:8280/ab"/>
```

This URL supports an exit point to ContactManager from the browser in which ClaimCenter is running. For example, the **Edit in ContactManager** button available when editing a contact uses this URL to open ContactManager. If you do not set this URL, the system uses the ContactManager URL defined in `suite-config.xml`. However, in that case, you cannot have separate browser and integration support for the URL.

7. In the **Project** window, expand **configuration**→**config**→**Plugins**→**registry**.
8. Double-click `ContactSystemPlugin.gwp` to open it in the Registry editor.
By default, the system uses the plugin implementation `gw.plugin.addressbook.demo.DemoContactSystemPlugin`. This Java plugin implementation is provided only for product demonstrations and is not to be used for any other purpose. In the steps that follow, you replace this class with the plugin implementation that connects ClaimCenter with ContactManager.
9. In the Registry editor, click Remove Plugin .
10. Click Add Plugin  and, in the drop-down menu, choose **Add Gosu Plugin** to register the new plugin implementation.

11. In the **Gosu Class** field, enter the following class:

```
gw.plugin.contact.ab1000.ABContactSystemPlugin
```


12. If necessary, start ContactManager. At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb runServer
```

13. In the Guidewire Studio for ClaimCenter **Project** window, navigate to **configuration**→**gsrc** and then to `wsi.remote.gw.webservice.ab.ab1000.wsc`.
14. Double-click the `ab1000.wsc` web services collection to open the editor, and then, in the editor, click the following resource:

```
${ab}/ws/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI?wsdl
```

The `${ab}` variable in this path corresponds to the `product name="ab"` entry in the `suite-config.xml` file that specifies the URL for ContactManager.

15. With ContactManager running, click **Fetch**  to get the latest updates to ABContactAPI.
16. In the same editor, look at the **Settings** tab.
There is a setting for the configuration provider `wsi.remote.gw.webservice.ab.ABConfigurationProvider`. This class defines the user name and password that ClaimCenter uses to authenticate with ContactManager when ClaimCenter makes calls to ABContactAPI. In the base configuration, `ABConfigurationProvider` defines the following user that is available in the ContactManager sample data.

```
config.Guidewire.Authentication.Username = "ClientAppCC"
config.Guidewire.Authentication.Password = "gw"
```

Note: Guidewire recommends that you change this user name and password in the `ABConfigurationProvider` class and in ContactManager.

17. Close Guidewire Studio for ClaimCenter.

Next steps

“Integrate ContactManager with ClaimCenter” on page 62

Integrate ContactManager with ClaimCenter

To set up the Guidewire ContactManager™ side of integration with Guidewire ClaimCenter™, use Guidewire Studio™ for ContactManager to change suite and plugin registry settings.

Before you begin

Complete “Integrate ClaimCenter with ContactManager” on page 61.

Procedure

1. At a command prompt, navigate to the ContactManager installation folder, and then start Guidewire Studio™ for ContactManager by entering the following command:

```
gwb studio
```




2. In Guidewire Studio for ContactManager, open the **Project** window.
3. Press **Ctrl+Shift+N** and enter `suite-config.xml` to find this file, and then double-click the search result to open the file in the editor.

This file defines the URLs of the applications in the Guidewire InsuranceSuite. If the entries in this file are commented out, you must remove the comments for the applications you have installed. In the default configuration, the Guidewire applications have the following settings in this file:

```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <!--<product name="cc" url="http://localhost:8080/cc"/>-->
  <!--<product name="pc" url="http://localhost:8180/pc"/>-->
  <!--<product name="ab" url="http://localhost:8280/ab"/>-->
  <!--<product name="bc" url="http://localhost:8580/bc"/>-->
</suite-config>
```

4. Remove the comments around the ClaimCenter (cc) entry, as follows:

```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <product name="cc" url="http://localhost:8080/cc"/>
  <!--<product name="pc" url="http://localhost:8180/pc"/>-->
  <!--<product name="ab" url="http://localhost:8280/ab"/>-->
  <!--<product name="bc" url="http://localhost:8580/bc"/>-->
</suite-config>
```

5. Open the **Project** window and expand **configuration**→**config**→**Plugins**→**registry**.
6. Double-click **ClaimSystemPlugin.gwp** to open it in the Plugin Registry editor.
You need to register a plugin implementation so ContactManager can use it to synchronize address book changes with ClaimCenter. By default, the system uses `gw.plugin.integration.StandAloneClientSystemPlugin`. You must replace this plugin implementation because it does not broadcast Contact changes and does not communicate with core applications.
7. In the Registry editor, click Remove Plugin .
8. Click Add Plugin  and, in the drop-down menu, choose **Add Gosu Plugin** to register the new plugin implementation.
9. In the **Gosu Class** field, enter the following class:
`gw.plugin.claim.cc1000.CCClaimSystemPlugin`
10. Navigate to **configuration**→**gsrsc** and then to `wsi.remote.gw.webservice.cc.cc1000.wsc`.
11. Double-click the `cc1000.wsc` web services collection to open the editor, and then, in the editor, click the following resource: `${cc}/ws/gw/webservice/cc/cc1000/contact/ContactAPI?wsdl`.
ContactManager calls this ClaimCenter webservice when it has Contact updates for ClaimCenter. The ClaimCenter web service `ContactAPI` implements the interface `ABClientAPI`. The `${cc}` variable in this path corresponds to the `product name="cc"` entry in the `suite-config.xml` file, which specifies the URL for ClaimCenter.
12. Ensure that the ClaimCenter server is running, and then click **Fetch**  to get the latest updates to `ContactAPI`.
13. In the same editor on the **Settings** tab is the configuration provider class. This class, `wsi.remote.gw.webservice.cc.CCConfigurationProvider`, defines the user name and password that ContactManager must use to connect with ClaimCenter.

IMPORTANT By default, ContactManager uses the user name `su` and password `gw`. Guidewire recommends that you change this user name and password in ClaimCenter and update `CCConfigurationProvider` in ContactManager.

14. Close Guidewire Studio for ContactManager.
15. To pick up your changes, stop and restart the two servers.
 - a. Stop ClaimCenter. Open a command prompt in the ClaimCenter installation folder and then enter the following command:

```
gwb stopServer
```

- b. Stop ContactManager. Open a command prompt in the ContactManager installation folder and then enter the following command:

```
gwb stopServer
```

- c. Start the ContactManager application in the ContactManager installation folder:

```
gwb runServer
```

- d. Start the ClaimCenter application in the ClaimCenter installation folder:

```
gwb runServer
```

Next steps

“Test the integration of ClaimCenter and ContactManager” on page 64

Test the integration of ClaimCenter and ContactManager

After setting up integration for ClaimCenter and ContactManager, you can perform some tests to ensure that the integration is working.

Before you begin

Complete “Integrate ContactManager with ClaimCenter” on page 62. Additionally, ensure that both ContactManager and ClaimCenter have started. Finally, if you are using the base configurations of ClaimCenter and ContactManager, ContactManager must have a ClientAppCC user for ClaimCenter to authenticate with. Sample data in ContactManager includes this user.

Procedure

1. When ClaimCenter is ready, open a browser window and enter the ClaimCenter URL. For example:

```
http://localhost:8080/cc/
```

2. Log in as a user who has permissions to view, create, edit, and search for a ContactManager contact in ClaimCenter. For example, log in as the sample user `ssmith` with password `gw`.
3. Open an existing claim and then click **Parties Involved** to open the **Contacts** screen.
4. Click **New Contact** and create a new contact.
5. Give the contact enough data to be able to save it in ContactManager, such as name, address, and tax ID information.
6. Give the contact a role on the claim and click **Update** to save it.
7. When the **Contacts** screen opens, select the new contact. Because you are logged in with a role that permits writing to ContactManager, the contact links automatically to ContactManager. If the link is successful, you see the message, *This contact is linked to the Address Book and is in sync.*
8. Click the **Address Book** tab to search for a contact.

With sample data loaded in ContactManager, you can search for a Vendor (Company) with a name that starts with the letter `a`. That search returns contacts with names like `AB Construction` and `Allendale, Myers & Associates`.
9. Open a browser window and enter the following ContactManager URL:

```
http://localhost:8280/ab/
```

10. Log in as a user who can view or edit a contact in ContactManager, such as the sample user `aaggregate` with password `gw`.
11. In ContactManager, click **Search** and verify that you can search for and locate the contact you just created in ClaimCenter.
12. Click the contact in the search results and edit it. For example, change the contact’s phone number. Click **Update** to save the changes.
13. In ClaimCenter, open the claim to which you added the new contact, and click **Parties Involved** to open the **Contacts** screen.

14. Select the contact that you added.

On the **Basics** card, if you have not changed default settings, you see the following message:

This contact is linked to the Address Book and is in sync.

15. Click **View in Address book**.

ClaimCenter fetches the contact data from ContactManager and displays it in a new screen.

- a. In this screen, you can click **Edit in ContactManager** to open a browser window and connect to the ContactManager server.

The system looks for the ContactManager URL in the configuration parameter `ContactSystemURL`, or in `suite-config.xml` if that parameter is not defined. You must also have a ContactManager user name to be able to use this feature. Additionally, that user must have a role in ContactManager that enables editing contacts, such as the Contact Manager role.

- b. If necessary, log in to ContactManager.
- c. Edit the contact data directly.
- d. Go back to the ClaimCenter window in your browser.

16. Click **Return to Contacts** near the top of the screen.

You see the **Contacts** screen again, with your contact on the **Basics** card showing the changes you made in ContactManager.

Next steps

If you encounter errors during testing, see “Troubleshooting the ClaimCenter connection with ContactManager” on page 65.

Troubleshooting the ClaimCenter connection with ContactManager

If you encounter errors when you test the ClaimCenter integration with ContactManager, you can troubleshoot them.

See also

- “Test the integration of ClaimCenter and ContactManager” on page 64.

ClaimCenter error: bad username or password

During testing of the ClaimCenter connection with ContactManager, you might see an error message saying that there was an exception caused by a bad user name or password. If so, you probably need to create the ContactManager user that ClaimCenter uses to communicate with ContactManager. In the base configuration, this user is `ClientAppCC` with password `gw`. This user is available if you import the sample data for ContactManager.

See also

- “Configuring core application authentication with ContactManager” on page 78
- For instructions on how to load sample data for ContactManager, see “Load sample data for ContactManager” on page 56
- To create a user in ContactManager, see “Configuring ContactManager authentication with core applications” on page 82

ClaimCenter error: no response from ContactManager

During testing of the ClaimCenter connection with ContactManager, you might not be able to get any response at all from ContactManager. If so, it is possible that the message queue that ClaimCenter uses to communicate with ContactManager is suspended. In that case, you must reactivate the ClaimCenter message queue.

See also

- “Reactivate the ClaimCenter message queue” on page 66

Reactivate the ClaimCenter message queue

If you are not getting a response from ContactManager in ClaimCenter, you can restart the message queue that ClaimCenter uses to communicate with ContactManager.

Before you begin

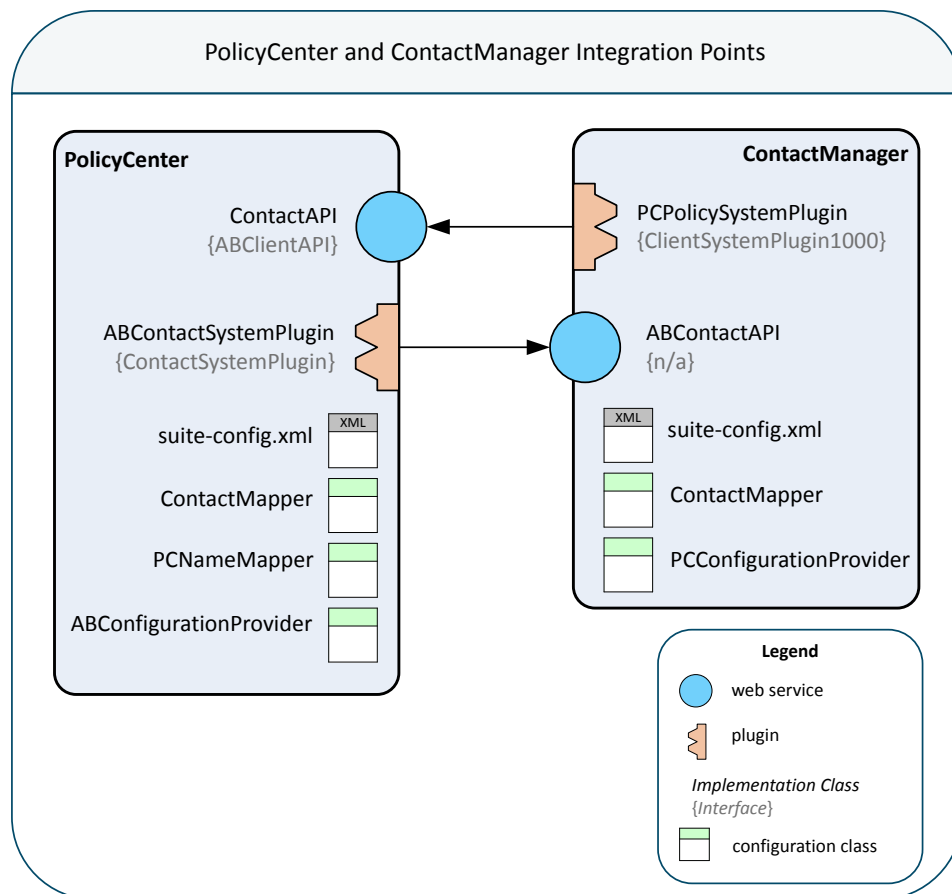
Integrate ContactManager and ClaimCenter.

Procedure

1. Ensure that ClaimCenter and ContactManager are running.
2. Log in to ClaimCenter as a user with administrative privileges. For example log in with user name `su` and password `gw`.
3. Click the **Administration** tab, and then navigate to **Monitoring**→**Message Queues** in the Sidebar.
4. On the **Message Queues** screen, select the check box next to the **Contact Message Transport** entry.
5. If the **Status** is **Suspended**, click the **Resume** button at the top of the screen.
6. The **Status** changes to **Started**, and ClaimCenter can now send messages to ContactManager.

Integrating ContactManager with PolicyCenter in QuickStart

The following figure shows the primary integration points used to integrate PolicyCenter and ContactManager.



In general, to get the basic integration working, you edit the `suite-config.xml` files in PolicyCenter and ContactManager. You also register the `PCPolicySystemPlugin` plugin implementation in the ContactManager plugin registry `PolicySystemPlugin.gwp`. Additionally, you register `ABContactSystemPlugin` in the PolicyCenter registry `ContactSystemPlugin.gwp`.

If you extend the contact model, you must edit the `ContactMapper` classes in both `ContactManager` and `PolicyCenter`, and possibly the `PCNameMapper` class.

IMPORTANT Guidewire recommends that you change the authentication user and password. To do so, you add authentication users to `ContactManager` and `ClaimCenter` and edit the `ABConfigurationProvider` and `CCConfigurationProvider` classes.

Integrating `ContactManager` and `PolicyCenter` is a multi-step process:

1. “Integrate `PolicyCenter` with `ContactManager`” on page 67
2. “Integrate `ContactManager` with `PolicyCenter`” on page 69
3. “Test the integration of `PolicyCenter` and `ContactManager`” on page 71
4. “Troubleshooting the `PolicyCenter` connection with `ContactManager`” on page 72

See also

- “Configure `PolicyCenter`-to-`ContactManager` authentication” on page 80
- “Configure `ContactManager`-to-`PolicyCenter` authentication” on page 85

Integrate `PolicyCenter` with `ContactManager`

This step is the first in the multi-step process of integrating `PolicyCenter` and `ContactManager`.

About this task

This step is the first in the multi-step process of integrating `ContactManager` and `PolicyCenter`. For an overview of the integration process, see “Integrating `ContactManager` with `PolicyCenter` in QuickStart” on page 66.

Procedure

1. At a command prompt, navigate to the `PolicyCenter` installation folder, and then start Guidewire Studio™ for `PolicyCenter` by entering the following command:

```
gwb studio
```

2. In Guidewire Studio for `PolicyCenter`, open the **Project** window.
3. Press **Ctrl+Shift+N** and enter `suite-config.xml` to find this file, and then double-click the search result to open the file in the editor.

This file defines the URLs of the applications in the Guidewire InsuranceSuite. If the entries in this file are commented out, you must remove the comments for the applications you have installed. In the default configuration, the Guidewire applications have the following settings in this file:



```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <!--<product name="cc" url="http://localhost:8080/cc"/>-->
  <!--<product name="pc" url="http://localhost:8180/pc"/>-->
  <!--<product name="ab" url="http://localhost:8280/ab"/>-->
  <!--<product name="bc" url="http://localhost:8580/bc"/>-->
</suite-config>
```

4. Remove the comments around the `ContactManager` (ab) entry, as follows:

```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <!--<product name="cc" url="http://localhost:8080/cc"/>-->
  <!--<product name="pc" url="http://localhost:8180/pc"/>-->
  <product name="ab" url="http://localhost:8280/ab"/>
  <!--<product name="bc" url="http://localhost:8580/bc"/>-->
</suite-config>
```

5. In the **Project** window, expand **configuration**→**config**→**Plugins**→**registry**.
6. Double-click `ContactSystemPlugin.gwp` to change the plugin implementation used by the system to connect to `ContactManager`.

By default, the system uses `gw.plugin.contact.impl.StandAloneContactSystemPlugin`. PolicyCenter uses this plugin when not connected to a contact management system. In the steps that follow, you replace this class with the plugin implementation that connects PolicyCenter with ContactManager.

7. In the Registry editor, click Remove Plugin .
8. Click Add Plugin  and, in the drop-down menu, choose **Add Gosu Plugin** to register the new plugin implementation.
9. In the **Gosu Class** field, enter the following class:
`gw.plugin.contact.ab1000.ABContactSystemPlugin`
10. If necessary, start ContactManager.
At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb runServer
```

11. In the Guidewire Studio for PolicyCenter **Project** window, navigate to **configuration**→**gsrsc** and then to `wsi.remote.gw.webservice.ab.ab1000.wsc`.
12. Double-click the `ab1000.wsc` web services collection to open the editor, and then, in the editor, click the following resource:

```
${ab}/ws/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI?wsdl
```

The `${ab}` variable in this path corresponds to the product `name="ab"` entry in the `suite-config.xml` file that specifies the URL for ContactManager.

13. With ContactManager running, click **Fetch**  to get the latest updates to ABContactAPI.
14. In the same editor, look at the **Settings** tab.

There is a setting for the configuration provider `wsi.remote.gw.webservice.ab.ABConfigurationProvider`. This class defines the user name and password that PolicyCenter uses to authenticate with ContactManager when PolicyCenter makes calls to ABContactAPI. In the base configuration, ABConfigurationProvider defines a user provided in the ContactManager sample data.

```
config.Guidewire.Authentication.Username = "ClientAppPC"
config.Guidewire.Authentication.Password = "gw"
```

IMPORTANT Guidewire recommends that you change this user name and password in the ABConfigurationProvider class and in ContactManager.

15. In the **Project** window, press `Ctrl+N` and enter `ContactMessageTransport`. In the search results, double-click `ContactMessageTransport` to open this Gosu class in the editor.
16. Press `Ctrl+F` and enter `getAdminUserForIntegrationHandling` to find that method in the class. The method is defined as follows:

```
private function getAdminUserForIntegrationHandling() : User {
    return PLDependenciesGateway.getUserFinder().findByCredentialName("admin")
}
```

This method gets the PolicyCenter user that is assigned an activity if ContactManager throws an unhandled exception during communication with PolicyCenter. In response to the activity, this user reviews the contact's data and makes the needed corrections. After the user updates the contact, PolicyCenter sends it to ContactManager again.

- a. You must designate a user of your own to handle this activity.

The `getAdminUserForIntegrationHandling` method retrieves the user by login name. In the base configuration, this PolicyCenter user has all permissions. The generic text for the activity is:

Subject: Failed to add the contact '{Contact}' to the CMS.
 Description: An unexpected error occurred when adding the contact to the contact management system\:
 {Error Message}

- b. Assign the user roles with permissions that enable working with contacts.
 The contact and tag permissions in the base configuration are `abcreate`, `ctccreate`, `anytagcreate`, `abdelete`, `ctcdelete`, `anytagdelete`, `abedit`, `ctcedit`, `anytagedit`, `abview`, `abviewsearch`, `ctcview`, and `anytagview`.
- c. When you determine the user who will handle these activities, you can create your own class and copy the code in the `ContactMessageTransport` class into your class.
- d. In your class's `getAdminUserForIntegrationHandling` method, replace the parameter `admin` with the login name of your user.
- e. Navigate to **configuration→config→Plugins→registry** and register your class in the plugin registry `ContactMessageTransport.gwp`.

17. Close Guidewire Studio for PolicyCenter.

Next steps

“Integrate ContactManager with PolicyCenter” on page 69

Integrate ContactManager with PolicyCenter

To set up the Guidewire ContactManager™ side of integration with Guidewire PolicyCenter™, use Guidewire Studio™ for ContactManager to change suite and plugin registry settings.

Before you begin

Complete “Integrate PolicyCenter with ContactManager” on page 67.

Procedure

1. At a command prompt, navigate to the ContactManager installation folder, and then start Guidewire Studio™ for ContactManager by entering the following command:

```
gwb studio
```

2. In Guidewire Studio for ContactManager, open the **Project** window.
3. Press **Ctrl+Shift+N** and enter `suite-config.xml` to find this file, and then double-click the search result to open the file in the editor.

This file defines the URLs of the applications in the Guidewire InsuranceSuite. If the entries in this file are commented out, you must remove the comments for the applications you have installed. In the default configuration, the Guidewire applications have the following settings in this file:



```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <!--<product name="cc" url="http://localhost:8080/cc"/>-->
  <!--<product name="pc" url="http://localhost:8180/pc"/>-->
  <!--<product name="ab" url="http://localhost:8280/ab"/>-->
  <!--<product name="bc" url="http://localhost:8580/bc"/>-->
</suite-config>
```

4. Remove the comments around the PolicyCenter (pc) entry, as follows:

```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <!--<product name="cc" url="http://localhost:8080/cc"/>-->
  <product name="pc" url="http://localhost:8180/pc"/>
  <!--<product name="ab" url="http://localhost:8280/ab"/>-->
  <!--<product name="bc" url="http://localhost:8580/bc"/>-->
</suite-config>
```


5. Open the **Project** window and expand **configuration→config→Plugins→registry**.
6. Double-click `PolicySystemPlugin.gwp` to open it in the Plugin Registry editor.

You need to register a plugin implementation so ContactManager can use it to synchronize address book changes with PolicyCenter. By default, the system uses `gw.plugin.integration.StandAloneClientSystemPlugin`. You must replace this plugin implementation because it does not broadcast Contact changes and does not communicate with core applications.

7. In the Registry editor, click Remove Plugin .
8. Click Add Plugin  and, in the drop-down menu, choose **Add Gosu Plugin** to register the new plugin implementation.
9. In the **Gosu Class** field, enter the following class:
`gw.plugin.policy.pc1000.PCPolicySystemPlugin`
10. Navigate to **configuration** → **gsrsrc** and then to `wsi.remote.gw.webservice.pc.pc1000.wsc`.
11. Double-click the `pc1000.wsc` web services collection to open the editor, and then, in the editor, click the following resource:

```
${pc}/ws/gw/webservice/pc/pc1000/contact/ContactAPI?wsdl
```

ContactManager calls this PolicyCenter webservice when it has Contact updates for PolicyCenter. The PolicyCenter web service `ContactAPI` implements the interface `ABClientAPI`. The `${pc}` variable in this path corresponds to the `product name="pc"` entry in the `suite-config.xml` file, which specifies the URL for PolicyCenter.

12. Ensure that the PolicyCenter server is running, and then click **Fetch**  to get the latest updates to `ContactAPI`.
13. At the bottom of the window is the **Settings** tab. The default setting in this tab is the configuration provider class. This class, `wsi.remote.gw.webservice.pc.PCConfigurationProvider`, defines the user name and password that ContactManager uses to connect with PolicyCenter.

IMPORTANT In the base configuration, ContactManager uses the user name `su` and password `gw`. Guidewire recommends that you change this user name and password in the `PCConfigurationProvider` class and in ContactManager.

14. When the update finishes, close Guidewire Studio for ContactManager.
15. To pick up your changes, stop and restart ContactManager and PolicyCenter.

- a. Stop PolicyCenter.

Open a command prompt in the PolicyCenter installation folder and then enter the following command:

```
gwb stopServer
```

- b. Stop ContactManager.

Open a command prompt in the ContactManager installation folder and then enter the following command:

```
gwb stopServer
```

- c. Start the ContactManager application in the ContactManager installation folder:

```
gwb runServer
```

- d. Start the PolicyCenter application in the PolicyCenter installation folder:

```
gwb runServer
```

Next steps

“Test the integration of PolicyCenter and ContactManager” on page 71

Test the integration of PolicyCenter and ContactManager

After setting up integration for PolicyCenter and ContactManager, you can perform some tests to ensure that the integration is working.

Before you begin

You must complete “Integrate ContactManager with PolicyCenter” on page 69 before continuing with this step. Additionally, ensure that both ContactManager and PolicyCenter have started. Finally, if you are using the base configurations of PolicyCenter and ContactManager, ContactManager must have a ClientAppPC user for PolicyCenter to authenticate with. Sample data in ContactManager includes this user.

About this task

Verify that the two applications are integrated.

Procedure

1. When PolicyCenter is ready, open a browser window and enter the following PolicyCenter URL:

```
http://localhost:8180/pc/
```

2. Log in as a user who can create a contact in PolicyCenter, such as the sample user **aapplegate** with password **gw**.
3. Open an existing account from the **Account** tab.
4. Click **Contacts** in the left Sidebar menu.
5. At the top of the **Account File Contacts** screen, click **Create New Contact**.
6. Click **Additional Interest**→**From Address Book**.
7. Choose **Person** for the type and search for a contact that is stored in ContactManager.
With sample data loaded in ContactManager, you could search for the person William Andy.
8. In the search results, look for a contact that has its **External** column set to **Yes**. That contact is stored only in ContactManager.
If a contact is stored in both PolicyCenter and ContactManager, its **External** column is set to **No**. If you do not see any contacts that are external, either ContactManager has not found a contact stored only in ContactManager or it has suspended its PolicyCenter message queue.
 - If you are sure the contact is stored only in ContactManager, the contact’s tag might not be set to Client. Any contact created in PolicyCenter and stored in ContactManager automatically gets a Client tag, but you have to add the tag manually if you create the contact directly in ContactManager.
 - An additional possibility is that PolicyCenter has stopped the message queue it uses for ContactManager.
9. Click **Select** for the person found in ContactManager.
You see that contact added to the list of account file contacts in the role Additional Interest.
10. Open a browser window and enter the following ContactManager URL:
`http://localhost:8280/ab/`
11. Log in as a user who can view or edit a contact in ContactManager, such as the sample user **aapplegate** with password **gw**.
12. In ContactManager, click **Search** and find the contact you added to the account in PolicyCenter.
13. Edit that contact and change the address, and then update the contact.
14. In PolicyCenter, go to the same **Contact** screen for the account to which you added the contact, and then click the contact.
15. Verify that the contact’s address has changed to the one you specified in ContactManager.

Next steps

If you encounter errors during testing, see “Troubleshooting the PolicyCenter connection with ContactManager” on page 72.

Troubleshooting the PolicyCenter connection with ContactManager

If you encounter errors when you test the PolicyCenter integration with ContactManager, you can troubleshoot them.

See also

- “Test the integration of PolicyCenter and ContactManager” on page 71.

PolicyCenter connection with ContactManager not working

You must be working with either a policy that is in force or an account that has at least one policy in force for contacts to be saved in ContactManager. If PolicyCenter appears not to be working with ContactManager and there is no error message, check the policy or account status.

PolicyCenter error: bad username or password

You might see an error message saying that there was an exception caused by a bad user name or password. If so, you probably need to create the ContactManager user that PolicyCenter uses to communicate with ContactManager. In the base configuration, this user is ClientAppPC with password gw. This user is available if you import the sample data for ContactManager.

See also

- To create a user in ContactManager, see “Configuring ContactManager authentication with core applications” on page 82.
- “Configuring core application authentication with ContactManager” on page 78.
- “Load sample data for ContactManager” on page 56.

PolicyCenter error: no response from ContactManager

If you are not able to get any response at all from ContactManager, it is possible that the message queue that PolicyCenter uses to communicate with ContactManager has been suspended.

See also

- “Reactivate the PolicyCenter message queue” on page 72

Reactivate the PolicyCenter message queue

If you are not getting a response from ContactManager in PolicyCenter, you can restart the message queue that PolicyCenter uses to communicate with ContactManager.

Before you begin

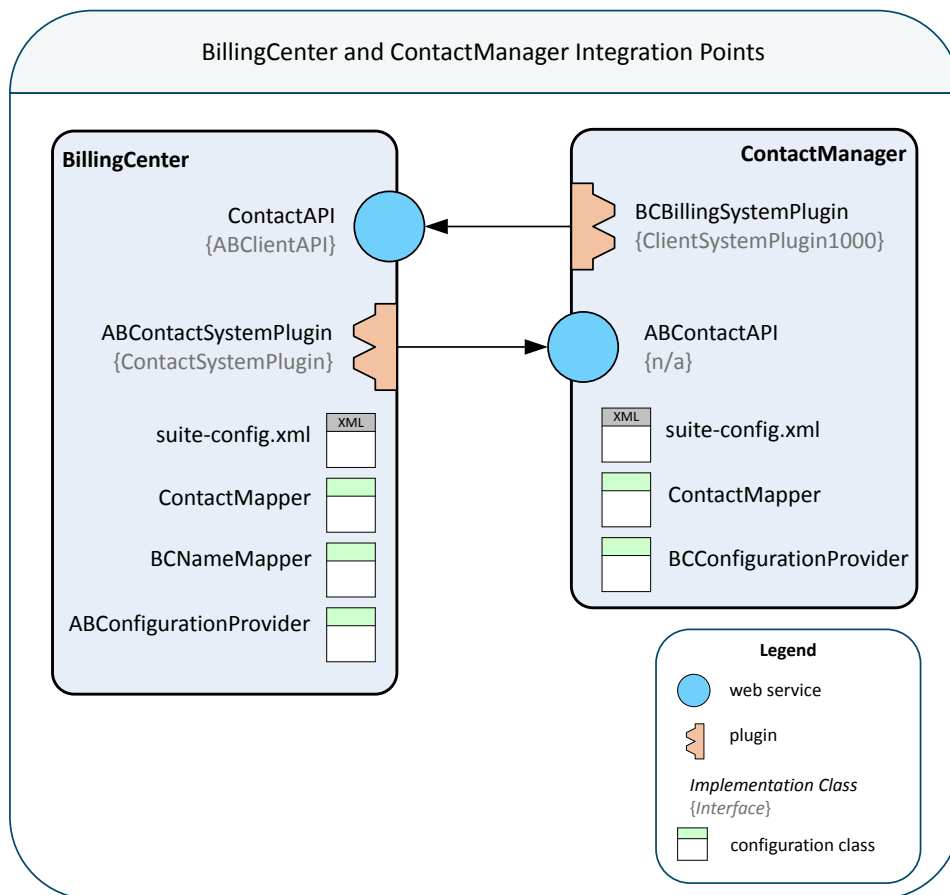
Integrate ContactManager and PolicyCenter.

Procedure

1. If they are not already running, start PolicyCenter and ContactManager.
2. Log in to PolicyCenter as a user with administrative privileges. For example, log in with user name su and password gw.
3. Click the **Administration** tab, and then navigate to **Monitoring**→**Message Queues** in the Sidebar on the left.
4. On the **Message Queues** screen, select the **ContactMessageTransport** entry.
5. If the **Status** is **Suspended**, click the **Resume** button at the top of the screen.
6. The **Status** changes to **Started**, and PolicyCenter can now send messages to ContactManager.

Integrating ContactManager with BillingCenter in QuickStart

The following figure shows the primary integration points used to integrate BillingCenter and ContactManager.



In general, to get the basic integration working, you edit the `suite-config.xml` files in BillingCenter and ContactManager. You also register the `BCBillingSystemPlugin` plugin implementation in the ContactManager plugin registry `BillingSystemPlugin.gwp`. Additionally, you register `ABContactSystemPlugin` in the BillingCenter registry `ContactSystemPlugin.gwp`.

If you extend the contact model, you edit the `ContactMapper` classes in both ContactManager and BillingCenter, and possibly the `BCNameMapper` class.

IMPORTANT Guidewire recommends that you change the authentication user and password. To do so, you add authentication users to ContactManager and ClaimCenter and edit the `ABConfigurationProvider` and `CCConfigurationProvider` classes.

Detailed steps for integrating BillingCenter and ContactManager are a multi-step process:

1. “Integrate BillingCenter with ContactManager” on page 73
2. “Integrate ContactManager with BillingCenter” on page 75
3. “Test the integration of BillingCenter and ContactManager” on page 77
4. “Troubleshooting the BillingCenter connection with ContactManager” on page 78

See also

- “Configure BillingCenter-to-ContactManager authentication” on page 81
- “Configure ContactManager-to-BillingCenter authentication” on page 87
- “Extending the client data model” on page 152

Integrate BillingCenter with ContactManager

This step is the first in the multi-step process of integrating BillingCenter and ContactManager.

About this task

For an overview of the integration process, see “Integrating ContactManager with BillingCenter in QuickStart” on page 72.

Procedure

1. At a command prompt, navigate to the BillingCenter installation folder, and then start Guidewire Studio™ for BillingCenter by entering the following command:

```
gwb studio
```



2. In Guidewire Studio for BillingCenter, open the **Project** window.
3. Press **Ctrl+Shift+N** and enter `suite-config.xml` to find this file, and then double-click the search result to open the file in the editor.

This file defines the URLs of the applications in the Guidewire InsuranceSuite. If the entries in this file are commented out, you must remove the comments for the applications you have installed. In the default configuration, the Guidewire applications have the following settings in this file:

```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <!--<product name="cc" url="http://localhost:8080/cc"/>-->
  <!--<product name="pc" url="http://localhost:8180/pc"/>-->
  <!--<product name="ab" url="http://localhost:8280/ab"/>-->
  <!--<product name="bc" url="http://localhost:8580/bc"/>-->
</suite-config>
```

4. Remove the comments around the ContactManager (ab) entry, as follows:

```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <!--<product name="cc" url="http://localhost:8080/cc"/>-->
  <!--<product name="pc" url="http://localhost:8180/pc"/>-->
  <product name="ab" url="http://localhost:8280/ab"/>
  <!--<product name="bc" url="http://localhost:8580/bc"/>-->
</suite-config>
```

5. In the **Project** window, expand **configuration**→**config**→**Plugins**→**registry**.
6. Double-click `ContactSystemPlugin.gwp` to open it in the Registry editor.
By default, the system uses the plugin implementation `gw.plugin.contact.impl.StandAloneContactSystemPlugin`. BillingCenter uses this Gosu plugin if it is not integrated with a contact management system. In the steps that follow, you replace this class with the plugin implementation that connects BillingCenter with ContactManager.
7. In the Registry editor, click **Remove Plugin** .
8. Click **Add Plugin**  and, in the drop-down menu, choose **Add Gosu Plugin** to register the new plugin implementation.
9. In the **Gosu Class** field, enter the following class:

```
gw.plugin.contact.ab1000.ABContactSystemPlugin
```

10. If necessary, start ContactManager.
At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb runServer
```

11. In the Guidewire Studio for BillingCenter **Project** window, navigate to **configuration**→**gsrc** and then to `wsi.remote.gw.webservice.ab.ab1000.wsc`.
12. Double-click the `ab1000.wsc` web services collection to open the editor, and then, in the editor, click the following resource:

```
${ab}/ws/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI?wsdl
```

The `${ab}` variable in this path corresponds to the `product name="ab"` entry in the `suite-config.xml` file that specifies the URL for ContactManager.

13. With ContactManager running, click **Fetch**  to get the latest updates to ABContactAPI.

14. In the same editor, look at the **Settings** tab.

There is a setting for the configuration provider `ws.remote.gw.webservice.ab.ABConfigurationProvider`. This class defines the user name and password that BillingCenter uses to authenticate with ContactManager when BillingCenter makes calls to ABContactAPI. In the base configuration, `ABConfigurationProvider` defines the following:

```
config.Guidewire.Authentication.Username = "su"
config.Guidewire.Authentication.Password = "gw"
```

Note: Guidewire recommends that you change this user name and password in the `ABConfigurationProvider` class and in ContactManager.

15. When the update finishes, close Guidewire Studio for BillingCenter.

Next steps

The next step is “Integrate ContactManager with BillingCenter” on page 75.

Integrate ContactManager with BillingCenter

To set up the ContactManager side of integration with BillingCenter, use Guidewire Studio for ContactManager to change suite and plugin registry settings.

Before you begin

Complete “Integrate BillingCenter with ContactManager” on page 73.

Procedure

1. At a command prompt, navigate to the ContactManager installation folder, and then start Guidewire Studio™ for ContactManager by entering the following command:

```
gwb studio
```

2. In Guidewire Studio for ContactManager, open the **Project** window.
3. Press **Ctrl+Shift+N** and enter `suite-config.xml` to find this file, and then double-click the search result to open the file in the editor.



This file defines the URLs of the applications in the Guidewire InsuranceSuite. If the entries in this file are commented out, you must remove the comments for the applications you have installed. In the default configuration, the Guidewire applications have the following settings in this file:

```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <!--<product name="cc" url="http://localhost:8080/cc"/>-->
  <!--<product name="pc" url="http://localhost:8180/pc"/>-->
  <!--<product name="ab" url="http://localhost:8280/ab"/>-->
  <!--<product name="bc" url="http://localhost:8580/bc"/>-->
</suite-config>
```

4. Remove the comments around the BillingCenter (bc) entry, as follows:

```
<suite-config xmlns="http://guidewire.com/config/suite/suite-config">
  <!--<product name="cc" url="http://localhost:8080/cc"/>-->
  <!--<product name="pc" url="http://localhost:8180/pc"/>-->
  <!--<product name="ab" url="http://localhost:8280/ab"/>-->
  <product name="bc" url="http://localhost:8580/bc"/>
</suite-config>
```

5. Open the **Project** window and navigate to **configuration→config→Plugins→registry**.


6. Double-click `BillingSystemPlugin.gwp` to open it in the Registry editor.
You need to register a plugin implementation so `ContactManager` can use it to synchronize address book changes with `BillingCenter`. By default, the system uses `gw.plugin.integration.StandAloneClientSystemPlugin`. You must replace this plugin implementation because it does not broadcast `Contact` changes and does not communicate with core applications.
7. In the Registry editor, click **Remove Plugin** .
8. Click **Add Plugin**  and, in the drop-down menu, choose **Add Gosu Plugin** to register the new plugin implementation.
9. In the **Gosu Class** field, enter the following class:

```
gw.plugin.billing.bc1000.BCBillingSystemPlugin
```

10. Navigate to **configuration**→**gsrc** and then to `wsi.remote.gw.webservice.bc.bc1000.wsc`.
11. Double-click the `bc1000.wsc` web services collection to open the editor, and then, in the editor, click the following resource:

```
${bc}/ws/gw/webservice/bc/bc1000/contact/ContactAPI?wsdl
```

`ContactManager` calls this `BillingCenter` web service when it has `Contact` updates for `BillingCenter`. The `BillingCenter` web service `ContactAPI` implements the interface `ABClientAPI`. The `${bc}` variable in this path corresponds to the product `name="bc"` entry in the `suite-config.xml` file, which specifies the URL for `BillingCenter`.

12. Ensure that `BillingCenter` is running, and then click **Fetch**  to get the latest updates to `ContactAPI`.

13. In the same editor on the **Settings** tab is the configuration provider class `wsi.remote.gw.webservice.bc.BCConfigurationProvider`.

This class defines the user name and password that `ContactManager` uses to connect with `BillingCenter`.

IMPORTANT In the base configuration, `ContactManager` uses the user name `su` and password `gw`. Guidewire recommends that you change this user name in the `BCConfigurationProvider` class and in `ContactManager`.

14. When the update finishes, close Guidewire Studio for `ContactManager`.
15. To pick up your changes, stop and restart `ContactManager` and `BillingCenter`.
 - a. Stop `BillingCenter`.
Open a command prompt in the `BillingCenter` installation folder and then enter the following command:

```
gwb stopServer
```

- b. Stop `ContactManager`.
Open a command prompt in the `ContactManager` installation folder and then enter the following command:

```
gwb stopServer
```

- c. Start the `ContactManager` application in the `ContactManager` installation folder:

```
gwb runServer
```

- d. Start the `BillingCenter` application in the `BillingCenter` installation folder:

```
gwb runServer
```

Next steps

“Test the integration of `BillingCenter` and `ContactManager`” on page 77

Test the integration of BillingCenter and ContactManager

After setting up integration for BillingCenter and ContactManager, you can perform some tests to ensure that the integration is working.

Before you begin

Complete “Integrate ContactManager with BillingCenter” on page 75. Additionally, ensure that both ContactManager and BillingCenter have started.

About this task

Verify that the two applications are integrated. If you are using the base configurations of BillingCenter and ContactManager, ContactManager provides the su user for BillingCenter to authenticate with.

Procedure

1. When BillingCenter is ready, open a browser window and enter the following BillingCenter URL:

```
http://localhost:8580/bc/
```

2. Log in as a user who can create a contact in BillingCenter, such as the sample user `aaggregate` with password `gw`.
3. Open an existing account from the **Account** tab.
4. Click **Contacts** in the left Sidebar under **Actions**.
5. On the **Contacts** screen, click the **Edit** button.
6. Click **Add Existing Contact**, and search for a company you know is in ContactManager and has a Client tag, such as the sample company Albertson's.
If a contact is stored in both BillingCenter and ContactManager, its **External** column is set to **No**. If you do not see any contacts that are external, either ContactManager has not found contacts stored only in ContactManager or it has suspended its BillingCenter message queue.
 - If you are sure the contact is stored only in ContactManager, the contact's tag might not be set to Client. Any contact created in BillingCenter and stored in ContactManager does have a Client tag.
 - An additional possibility is that BillingCenter has stopped the message queue it uses for ContactManager.
7. Click **Select** next to the company name.
8. On the **Contacts** screen, select the company name.
9. Below, on the **Contact Info** card, change contact data, like the **Address 1** field, and then click **Update** to save the change.
10. Open a browser window and enter the following ContactManager URL:

```
http://localhost:8280/ab/
```

11. Log in as a user who can view or edit a contact in ContactManager, such as the sample user `aaggregate` with password `gw`.
12. In ContactManager, click **Search** and find the contact you changed in BillingCenter.
13. Verify that the data you changed for the contact in BillingCenter was also changed for the contact in ContactManager.

Next steps

If you encounter errors during testing, see “Troubleshooting the BillingCenter connection with ContactManager” on page 78.

Troubleshooting the BillingCenter connection with ContactManager

If you encounter errors when you test the BillingCenter integration with ContactManager, you can troubleshoot them.

BillingCenter error: bad username or password error

You might see an error message saying that there was an exception caused by a bad user name or password. If so, you probably need to create the ContactManager user that BillingCenter uses to communicate with ContactManager. In the base configuration, this user is `su` with password `gw`.

See also

- “Configuring ContactManager authentication with core applications” on page 82.
- “Configuring core application authentication with ContactManager” on page 78
- “Load sample data for ContactManager” on page 56.

BillingCenter error: no response from ContactManager

If you are not able to get any response at all from ContactManager, it is possible that the message queue that BillingCenter uses to communicate with ContactManager has been suspended.

See also

- “Reactivate the BillingCenter message queue” on page 78

Reactivate the BillingCenter message queue

If you are not getting a response from ContactManager in BillingCenter, you can restart the message queue that BillingCenter uses to communicate with ContactManager.

Before you begin

Integrate ContactManager and BillingCenter.

Procedure

1. If they are not already running, start BillingCenter and ContactManager.
2. Log in to BillingCenter as a user with administrative privileges. For example, log in with user name `su` and password `gw`.
3. Click the **Administration** tab, and then navigate to **Monitoring**→**Message Queues** in the sidebar on the left.
4. On the **Message Queues** screen, select the **Contact Message Transport** entry.
5. If the **Status** is **Suspended**, click the **Resume** button at the top of the screen.
6. The **Status** changes to **Started**, and BillingCenter can now send messages to ContactManager.

Configuring core application authentication with ContactManager

When you integrate a Guidewire core application with ContactManager, you register a plugin implementation with the core application’s plugin registry. For example, to integrate ContactManager 10.0 with a core application of the same version, you register `gw.plugin.contact.ab1000.ABContactSystemPlugin` with the core application’s `ContactSystemPlugin.gwp` registry.

Each core application defines a user name and password that it uses to authenticate with ContactManager when the application uses its plugin to call `ABContactAPI` methods. The definition is in the class

`wsi.remote.gw.webservice.ab.ABConfigurationProvider`. For security purposes, Guidewire recommends that you define your own user names and passwords.

- ClaimCenter defines the two authentication parameters, `username` and `password`, as `ClientAppCC` and `gw`.
- PolicyCenter defines the two authentication parameters, `username` and `password`, as `ClientAppPC` and `gw`.
- BillingCenter defines the two authentication parameters, `username` and `password`, as `su` and `gw`.

The base ContactManager application's sample code contains users with names and passwords that match the default settings in the core applications. The sample code has users with names `ClientAppCC`, `ClientAppPC`, and `su`, all with password `gw`.

Guidewire recommends that you change the `username` and `password` that each application uses to authenticate with ContactManager when making calls to `ABContactAPI`. The process, which is generally the same for all applications, follows:

1. Set up a user and password in ContactManager to match the user name and password you define in the core application.
2. Define the user name and password in the core application's `ABConfigurationProvider` class.

Configure ClaimCenter-to-ContactManager authentication

Guidewire recommends that you change the user name and password that ClaimCenter uses to authenticate with ContactManager.

Procedure

1. Start ContactManager.

At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb runServer
```

2. Log in as a user who can create new ContactManager users.
For example, log in with user name `su` and password `gw`.
3. Click the **Administration** tab, and then navigate to **Actions**→**New User**.
4. Enter the following values:

Name	Value
First Name	CC
Last Name	NewAuthUser
Username	CCNewAuthUser
Password	Xc8899Lm
Confirm Password	Xc8899Lm
Active	Yes
Locked	No

5. Under **Roles**, click **Add**.
6. Click the **Name** field and choose **Client Application**.
7. Click **Update**.
8. Start Guidewire Studio for ClaimCenter.

At a command prompt, navigate to the ClaimCenter installation folder and enter the following command:


```
gwb studio
```

9. Open the **Project** window.
10. Press **Ctrl+N** and enter `ABConfigurationProvider`, and then click the class name in the search results to open it in the editor.

11. Change the Username and Password definition in the configure method.

For example, change the values to CCNewAuthUser and Xc8899Lm, as follows:

```
override function configure( serviceName : QName, portName : QName, config : Wsd1Config ) {
    config.Guidewire.Authentication.Username = "CCNewAuthUser"
    config.Guidewire.Authentication.Password = "Xc8899Lm"
}
```

12. In the **Project** window, navigate to **configuration**→**gsrc** and then to `wsi.remote.gw.webservice.ab.ab1000.wsc`.
13. Double-click the `ab1000.wsc` web resource collection to open the editor.
14. In the editor, click `${ab}/ws/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI?wsdl`.
15. With ContactManager still running, click **Fetch** .
16. When the update finishes, stop the ClaimCenter server if it is running, and then restart it to pick up this change.
17. Log in to ClaimCenter as a user with permission to work with the Address Book, such as user `ssmith` with password `gw`.
18. Verify that you can use the **Address Book** tab to search for ContactManager contacts, such as the sample contact William Weeks.

Configure PolicyCenter-to-ContactManager authentication

Guidewire recommends that you change the user name and password that PolicyCenter uses to authenticate with ContactManager.

Procedure

1. Start ContactManager.

At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb runServer
```

2. Log in as a user who can create new ContactManager users.
For example, log in with user name `su` and password `gw`.
3. Click the **Administration** tab, and then navigate to **Actions**→**New User**.
4. Enter the following values:

Name	Value
First Name	PC
Last Name	NewAuthUser
Username	PCNewAuthUser
Password	Xc8899Lm
Confirm Password	Xc8899Lm
Active	Yes
Locked	No


5. Under **Roles**, click **Add**.
6. Click the **Name** field and choose **Client Application**.
7. Click **Update**.
8. Start Guidewire Studio for PolicyCenter.

At a command prompt, navigate to the PolicyCenter installation folder and enter the following command:

```
gwb studio
```


9. Press Ctrl+N and enter ABConfigurationProvider, and then click the class name in the search results to open it in the editor.
10. Change the Username and Password definition in the configure method.
For example, change the values to PCNewAuthUser and Xc8899Lm, as follows:

```
override function configure( serviceName : QName, portName : QName, config : Wsd1Config ) {
    config.Guidewire.Authentication.Username = "PCNewAuthUser"
    config.Guidewire.Authentication.Password = "Xc8899Lm"
}
```

11. In the **Project** window, navigate to **configuration**→**gsrsc** and then to `wsi.remote.gw.webservice.ab.ab1000.wsc`.
12. Double-click the `ab1000.wsc` web services collection to open the editor.
13. In the editor, click `${ab}/ws/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI?wsdl`.
14. With ContactManager still running, click **Fetch** .
15. When the update finishes, stop the PolicyCenter server if it is running, and then restart it to pick up this change.
16. Log in to PolicyCenter as a user with permission to search for and change contacts, such as user name `aaplegate` with password `gw`.
17. Verify that you can use the **Contact** tab to search for ContactManager contacts. You can be sure that a contact is stored in ContactManager if its **External** field has the value **Yes**. For example, search for the person Adam Hinds, a client contact in the ContactManager sample data.

Configure BillingCenter-to-ContactManager authentication

Guidewire recommends that you change the user name and password that BillingCenter uses to authenticate with ContactManager.

Procedure

1. Start ContactManager.
At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb runServer
```

2. Log in as a user who can create new ContactManager users.
For example, log in with user name `su` and password `gw`.
3. Click the **Administration** tab, and then navigate to **Actions**→**New User**.
4. Enter the following values:

Name	Value
First Name	BC
Last Name	NewAuthUser
Username	BCNewAuthUser
Password	Xc8899Lm
Confirm Password	Xc8899Lm
Active	Yes
Locked	No

5. Under **Roles**, click **Add**.
6. Click the **Name** field and choose **Client Application**.
7. Click **Update**.

8. Start Guidewire Studio for BillingCenter.

At a command prompt, navigate to the BillingCenter installation folder and enter the following command:

```
gwb studio
```

9. Press Ctrl+N and enter ABConfigurationProvider, and then click the class name in the search results to open it in the editor.

10. Change the Username and Password definition in the configure method.

For example, change the values to BCNewAuthUser and Xc8899Lm, as follows:

```
override function configure( serviceName : QName, portName : QName, config : Wsd1Config ) {
    config.Guidewire.Authentication.Username = "BCNewAuthUser"
    config.Guidewire.Authentication.Password = "Xc8899Lm"
}
```

11. In the **Project** window, navigate to **configuration→gsrsc** and then to

wsi.remote.gw.webservice.ab.ab1000.wsc.

12. Double-click the ab1000.wsc web services collection to open the editor.

13. In the editor, click `${ab}/ws/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI?wsdl`.

14. With ContactManager still running, click **Fetch** .

For more information on the web service editor, see the *Configuration Guide*.

15. When the update finishes, stop BillingCenter server if it is running, and then restart it to pick up this change.

16. Log in to BillingCenter as a user with permission to search for and change contacts, such as user name su with password gw.

17. Verify that you can use the **Contact** tab to search for ContactManager contacts. You can be sure that a contact is stored in ContactManager if its **External** field has the value **Yes**. For example, search for the person Adam Hinds, a client contact in the ContactManager sample data.

Configuring ContactManager authentication with core applications

You can change the user names and passwords that ContactManager uses to send ABContact updates to the core applications. In the base configuration, ContactManager defines this user as su, a user that by default has all permissions to perform any action in a core application. Guidewire recommends for security reasons that you change this default setting to a user with permission only to create, update, and delete contacts.

Overview of ContactManager authentication configuration

Initially, to set up ContactManager to send ABContact updates to a Guidewire core application, you open Guidewire Studio for ContactManager and register a plugin with the core application's plugin registry. For example:

- To communicate with ClaimCenter 9.0, you register gw.plugin.claim.cc1000.CCClaimSystemPlugin with the plugin registry ClaimSystemPlugin.gwp.
- To communicate with PolicyCenter 9.0, you register gw.plugin.policy.pc1000.PCPolicySystemPlugin with the plugin registry PolicySystemPlugin.gwp.
- To communicate with BillingCenter 9.0, you register gw.plugin.billing.bc1000.BCBillingSystemPlugin with the plugin registry BillingSystemPlugin.gwp.

ContactManager uses these plugins to make calls to core application web services that implement ABClientAPI.

ContactManager defines core application authorization for each web service in a configuration provider class, which is associated with the web service in a web service collection. To see the web services that ContactManager uses with a core application, you open the editor for the web service collection.

ContactManager provides the following web service collections for each core application. Each web service collection enables you to open and edit a configuration provider class, where you can define the user name and password that ContactManager uses with the core application.

ClaimCenter

cc1000.wsc uses wsi.remote.gw.webservice.cc.CCConfigurationProvider.

PolicyCenter

pc1000.wsc uses wsi.remote.gw.webservice.pc.PCConfigurationProvider.

BillingCenter

bc1000.wsc uses wsi.remote.gw.webservice.bc.BCConfigurationProvider.

In general, you change the user name and password as follows:

1. Set up a user with a name and password in the core application that matches the user name and password you define in the configuration provider class in ContactManager.
2. Give the user a role with limited permissions, with at least the permissions to create, edit, and delete contacts and the permission SOAP administration.
3. In ContactManager, define the user name and password in the configuration provider class for the core application web service.

Configure ContactManager-to-ClaimCenter authentication

Guidewire recommends that you change the user name and password that ContactManager uses to authenticate with ClaimCenter.

Procedure

1. Start the ClaimCenter server.

At a command prompt, navigate to the ClaimCenter installation folder and enter the following command:

```
gwb runServer
```

2. Log in as a user who can create new ClaimCenter users and roles.
For example, log in with user name su and password gw.
3. Create a user role with all the permissions that enable working with contacts whose contact information is stored in ContactManager. In the base configuration, the minimum required permission codes are abcreate, abcreatepref, anytagcreate, ctccreate, abdelete, abdeletepref, anytagdelete, abedit, abeditpref, anytagedit, ctccedit, and soapadmin.
 - a. Click the **Administration** tab, and then navigate to **Actions→Roles**.
 - b. On the **Roles** screen, click **Add Role**.
 - c. For **Name** enter Contact Data Admin.
 - d. For **Type**, choose **User Role**.
 - e. For **Description** enter Permissions for virtual user required by ContactManager for updating contacts.
 - f. Beneath the **Description** field, click **Add**.
 - g. Click the **Permission** field and choose **Create address book contacts** from the list.

- h. Repeat the process for the following permissions. For each permission, click **Add** and choose the permission from the drop-down list:
 - Create address book preferred vendors
 - Create contact with any tag
 - Create local contacts
 - Delete address book contacts
 - Delete address book preferred vendors
 - Delete contact with any tag
 - Edit address book contacts
 - Edit address book preferred vendors
 - Edit contact with any tag
 - Edit local contacts
 - SOAP administration
 - i. Click **Update** to create the new Contact Data Admin role.
4. Choose **Actions**→**New User**.
 5. Enter the following values for a new user named CMUpdateCC:

Name	Value
First Name	CMUpdate
Last Name	CC
Username	CMUpdateCC
Password	Xc8899Lm
Confirm Password	Xc8899Lm
Active	Yes
Locked	No


6. Under **Roles**, click **Add**.
7. Click the **Name** field and choose **Contact Data Admin**.
 If this role has more permissions than you would like, you can create a new role and assign the new user to that role. The new role must have at least the following permissions: `abcreate`, `abdelete`, `abedit`, `abview`, `abcreatepref`, `abdeletepref`, `abeditpref`, `abviewsearch`, `anytagcreate`, `anytagdelete`, `anytagedit`, `anytagview`, `ctccreate`, `ctccedit`, `ctcview`, and `soapadmin`.
8. Click **Update**.
9. Start Guidewire Studio for ContactManager.
 At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb studio
```

10. In the **Project** window, navigate to **configuration**→**gsrsrc**, and then navigate to `wsi.remote.gw.webservice.cc.CCConfigurationProvider`.
11. Double-click `CCConfigurationProvider` to open it in the editor.
12. Change the Username and Password definition in the `configure` method and then save your changes.
 For example, define Username and Password to be `CMUpdateCC` and `Xc8899Lm`, as follows:

```
override function configure( serviceName : QName,
                           portName : QName,
                           config : WsdlConfig ) {
    config.Guidewire.Authentication.Username = "CMUpdateCC"
    config.Guidewire.Authentication.Password = "Xc8899Lm"
}
```

13. In the editor, the `cc1000.wsc` web services collection is at the same level as the `CCConfigurationProvider` class. Double-click `cc1000.wsc`.
Alternatively, you can press `Ctrl+Shift+N` and enter `cc1000`, and then double-click `cc1000.wsc` in the search results.
14. In the editor, select the following resource:


```
${cc}/ws/gw/webservice/cc/cc1000/contact/ContactAPI?wsdl
```
15. Ensure that the **Settings** tab has the **Setting Type** for **ConfigurationProvider** set to `wsi.remote.gw.webservice.cc.CCConfigurationProvider`, the class you just edited.
16. With ClaimCenter still running, above the **Settings** tab, click **Fetch**  to update the web service's WSDL file.
17. When the update finishes, stop the ContactManager server and then restart it to pick up this change.
18. Log in to ClaimCenter as a user with permission to work with the Address Book, such as user `ssmith` with password `gw`.
19. Add to a claim a contact that is stored in ContactManager.
For example, open a claim, click **Parties Involved**, and on the **Contacts** screen, click **Add Existing Contact**. Search for a contact that you know is stored in ContactManager, like the sample contact Stephen Marshall. Select the contact and give the contact a role on the claim, and then click **Update**.
20. Log in to ContactManager as a user who can work with contacts.
For example, log in as the sample user `aaplegate` with password `gw`.
21. Make a change to the contact that you added to the claim, such as a different phone number, and save it.
22. Back in ClaimCenter, on the **Contact** screen, select a different contact from the one you updated.
23. Select the contact that you added.
On the **Basics** card for that contact is the message, **This contact is linked to the Address Book but is out of sync**.
24. Click **Copy from Address Book** to update the contact.

Configure ContactManager-to-PolicyCenter authentication

Guidewire recommends that you change the user name and password that ContactManager uses to authenticate with PolicyCenter.

Procedure

1. Start the PolicyCenter server.
At a command prompt, navigate to the PolicyCenter installation folder and enter the following command:

```
gwb runServer
```

2. Log in as a user who can create new PolicyCenter users and roles.
For example, log in with user name `su` and password `gw`.
3. Create a user role with all the permissions that enable working with clients whose contact information is stored in ContactManager. The permission codes in the base configuration are `abcreate`, `ctccreate`, `anytagcreate`, `abdelete`, `anytagdelete`, `abedit`, `anytagedit`, `ctccedit`, and `soapadmin`.
 - a. Click the **Administration** tab, and then navigate to **Actions→Roles**.
 - b. On the **Roles** screen, click **New Role**.
 - c. For **Name** enter `Client Data Admin`.
 - d. For **Type**, choose **User Role**.
 - e. For **Description** enter `Permissions for virtual user required by ContactManager for updating contacts`.
 - f. Under **Permissions**, click **Add**.
 - g. Click the **Permission** field and choose **Create address book contacts** from the list.

- h. Repeat the process for the following permissions. For each permission, click **Add** and choose the permission from the drop-down list:
 - Create contact with any tag
 - Create local contacts
 - Delete address book contacts
 - Delete contact with any tag
 - Edit address book contacts
 - Edit contact with any tag
 - Edit local contacts
 - SOAP administration
- i. Click **Update** to create the new Client Data Admin role.
4. On the **Administration** tab, navigate to **Actions**→**New User**.
5. Enter the following values for a new user named CMUpdatePC:

Name	Value
First Name	CMUpdate
Last Name	PC
Username	CMUpdatePC
Password	Xc8899Lm
Confirm Password	Xc8899Lm
Active	Yes
Locked	No
Vacation Status	At work
User Type	Other
Primary Phone	Work
Work Phone	800-555-5555

Primary Phone and **Work Phone** are required for a new user, but they do not require real values for this particular user.


6. Click the **Roles** tab, and then click **Add**.
7. Click the **Name** field and choose Client Data Admin.
8. Click **Update**.
9. Start Guidewire Studio for ContactManager.
At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb studio
```

10. In the **Project** window, navigate to **configuration**→**gsrc**, and then navigate to `wsi.remote.gw.webservice.pc.PCConfigurationProvider`.
11. Double-click `PCConfigurationProvider` to open it in the editor.
12. Change the Username and Password definition in the `configure` method.
For example, define Username and Password to be `CMUpdatepc` and `Xc8899Lm`, as follows:

```
override function configure( serviceName : QName,
                           portName : QName,
                           config : WsdlConfig ) {
    config.Guidewire.Authentication.Username = "CMUpdatePC"
    config.Guidewire.Authentication.Password = "Xc8899Lm"
}
```

13. In the **Project** window, the `pc1000.wsc` web services collection is at the same level as the `PCConfigurationProvider` class. Double-click `pc1000.wsc`.
Alternatively, you can press `Ctrl+Shift+N` and enter `pc1000`, and then double-click `pc1000.wsc` in the search results.
14. In the editor, select the following resource:


```
${pc}/ws/gw/webservice/pc/pc1000/contact/ContactAPI?wsdl
```
15. Ensure that the **Settings** tab has the **Setting Type** for **ConfigurationProvider** set to `wsi.remote.gw.webservice.pc.PCConfigurationProvider`, the class you just edited.
16. With PolicyCenter still running, above the **Settings** tab, click **Fetch**  to update the web service's WSDL file.
17. When the update finishes, stop the ContactManager server and restart it to pick up this change.
18. Log in to PolicyCenter as a user with permission to work with the **Contact** tab and with accounts, such as the user `aaplegate` with password `gw`.
19. Add a contact stored in ContactManager to an active account. For example:
 - a. Open an active account, such as the sample data account `S000212121` named **Armstrong Cleaners**, and, in the Sidebar on the left under **Actions**, click **Contacts**.
The **Account File Contacts** screen opens.
 - b. Click **Create New Contact**→**Additional Insured**→**From Address Book**.
 - c. Search for a contact who has the Client tag set in ContactManager, such as the sample Person contact **Mark Stone**.
 - d. Click **Select** for a contact that the search results list says is stored in ContactManager.
The value of the contact's **External** field must be **Yes**.
 - e. PolicyCenter adds the selected contact to the list of contacts on the **Account File Contacts** screen.
20. Log in to ContactManager as a user who can work with contacts.
For example, log in as the sample user `aaplegate` with password `gw`.
21. Make a change to the contact that you added to the account, such as entering a different primary phone number, and then click **Update** to save the change.
22. Back in PolicyCenter, on the **Account File Contacts** screen, select a different contact from the one you added.
23. Select the contact that you added previously.
PolicyCenter updates the added contact's data.
24. Ensure that the data has changed for that contact.

Configure ContactManager-to-BillingCenter authentication

Guidewire recommends that you change the user name and password that ContactManager uses to authenticate with BillingCenter.

Procedure

1. Start the BillingCenter server.
At a command prompt, navigate to the BillingCenter installation folder and enter the following command:

```
gwb runServer
```

2. Log in as a user who can create new BillingCenter users.
For example, log in with user name `su` and password `gw`.
3. Create a user role with all the permissions that enable working with clients whose contact information is stored in ContactManager. The permission codes are `acctcntcreate`, `acctcntdelete`, `acctcntedit`, `plcycntcreate`, `plcycntdelete`, `plcycntedit`, `prodcntcreate`, `prodcntdelete`, `prodcntedit`, and `soapadmin`.
 - a. Click the **Administration** tab, and then navigate to **Actions**→**Roles**.

- b. On the **Roles** screen, click **New Role**.
 - c. For **Name** enter Client Data Admin.
 - d. For **Description** enter Permissions for virtual user required by ContactManager for updating contacts.
 - e. Below the **Description** field, click **Add**.
 - f. Click the **Permission** field and choose **Create account contact** from the list.
 - g. Repeat the process for the following permissions. For each permission, click **Add** and choose the permission from the drop-down list:
 - Create policy contact
 - Create producer contact
 - Delete account contact
 - Delete policy contact
 - Delete producer contact
 - Edit account contact
 - Edit policy contact
 - Edit producer contact
 - SOAP administration
 - h. Click **Update** to create the new Client Data Admin role.
4. On the **Administration** tab, navigate to **Actions**→**New User**.
 5. Enter the following values for a new user named CMUpdateBC:

Name	Value
First Name	CMUpdate
Last Name	BC
Username	CMUpdateBC
Password	Xc8899Lm
Confirm Password	Xc8899Lm

6. Click **Add User Role**.
7. Click the **Name** list and choose Client Data Admin.
8. Click **Next**. and add some fake data for the required fields **Address**, **City**, **Country**, **Primary Phone**, and the telephone number for the type of primary phone you selected.
9. Click **Next**.
10. Click **Finish**.
11. Start Guidewire Studio for ContactManager.
At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb studio
```


12. In the **Project** window, navigate to **configuration**→**gsrsrc**, and then navigate to `wsi.remote.gw.webservice.bc.BCConfigurationProvider`.
13. Double-click `BCConfigurationProvider` to open it in the editor.
14. Change the Username and Password definition in the `configure` method.
For example, define Username and Password to be CMUpdateBC and Xc8899Lm, as follows:

```
override function configure( serviceName : QName,
                           portName : QName,
                           config : WsdlConfig ) {
    config.Guidewire.Authentication.Username = "CMUpdateBC"
    config.Guidewire.Authentication.Password = "Xc8899Lm"
}
```


15. In the **Project** window, the `bc1000.wsc` web services collection is at the same level as the `BCConfigurationProvider` class. Double-click `bc1000.wsc`.
Alternatively, you can press `Ctrl+Shift+N` and enter `bc1000`, and then double-click `bc1000.wsc` in the search results.

16. In the editor, select the following resource:

```
${bc}/ws/gw/webservice/bc/bc1000/contact/ContactAPI?wsdl
```

17. Ensure that the **Settings** tab has the **Setting Type** for **ConfigurationProvider** set to `wsi.remote.gw.webservice.bc.BCConfigurationProvider`, the class you just edited.
18. With BillingCenter still running, above the **Settings** tab, click **Fetch**  to update the web service's WSDL file.
19. When the update finishes, stop the ContactManager server and then restart it to pick up this change.
20. Log in to BillingCenter as a user with permission to work with the **Contacts** screen for accounts, such as the user `su` with password `gw`.
21. Add a contact stored in ContactManager to an active account. For example:
 - a. Open an active account, such as the sample account named Standard Account, and, in the Sidebar on the left under **Actions**, click **Contacts**. The **Contacts** screen opens.
 - b. Click **Edit**, and then click **Add Existing Contact**.
 - c. Search for a contact that has the Client tag set in ContactManager, such as the sample Person contact Stan Newton. If necessary, log in to ContactManager and add a Client tag to a contact.
 - d. Click **Select** for a contact that the search results list says is stored in ContactManager.
The value of the contact's **External** field must be **Yes**.
 - e. BillingCenter adds the selected contact to the list of contacts on the **Contacts Edit** screen.
 - f. Click **Update** to add the contact to the account.
22. Log in to ContactManager as a user who can work with contacts.
For example, log in as the sample user `aaplegate` with password `gw`.
23. Make a change to the contact that you added to the account, such as entering a different address, and click **Update** to save the change.
24. Back in BillingCenter, on the **Contacts** screen, select the contact you added and ensure that the data has changed for that contact.

Searching for contacts

ContactManager supports search for contacts both in its own user interface and searches initiated externally through web services, such as from the Guidewire core applications. There are configuration points for contact search both in ContactManager and in the core applications.

Overview of contact search

Guidewire core applications store contacts locally, and, if integrated with ContactManager, a core application can also store contacts centrally in the ContactManager database. PolicyCenter and BillingCenter support searching for contacts both locally and in an external contact management application, like ContactManager. ClaimCenter supports searching for contacts only in an external contact management application, like ContactManager.

For example, in ClaimCenter, if you have added a contact to a claim, you can see a list of contacts by navigating to the **Parties Involved**→**Contacts** screen. All the contacts on the list are stored locally with the claim. Contacts can also be stored in ContactManager. If you click a contact in the list, you see its detailed information below the list. If the contact is stored in ContactManager, at the top of the **Basics** card is a message starting with the text **This contact is linked to the Address Book**.

- In ClaimCenter, you can search for a contact in the Address Book by clicking the **Address Book** tab and entering your search criteria. All contact data returned by this search comes from contacts stored in ContactManager. You can also search for contacts when creating existing contacts for claims. For example, with a claim open, you can navigate to the **Parties Involved**→**Contacts** screen and click **Add Existing Contact**.
- In PolicyCenter, you can search for a contact by clicking the **Contact** tab and then entering your search criteria. Alternatively, you can click the **Search** tab menu and choose **Contact**. Additionally, you can open the **Contacts** screen for an account or policy and select a contact. Then you can click **Add Existing Contact**, choose a contact type, and then choose **From Address Book** to open the contact search screen.
- In BillingCenter, you can search for a contact by clicking the **Search** tab and choosing **Contacts**, and then entering your search criteria. Additionally, if you edit the **Contacts** screen for an account or policy period, clicking **Add Existing Contact** opens the contact search screen.

Searching for contacts in PolicyCenter and BillingCenter returns information both about locally stored contacts and about contacts stored in an external contact management system, such as ContactManager. The list of search results has an **External** field that provides the following information:

- If the value in this field is **Yes**, the contact is not local, but is stored in an external contact management system.
- If the value in this field is **No**, the contact is stored locally and might also be stored in an external contact management system.

In all cases, searching for contacts returns a subset of information about the contact, which you see listed in the set of returned contacts.

IMPORTANT Limiting the fields retrieved by a search improves performance. In the base configuration, ContactManager filters search results to reduce the size of the objects returned to the calling application. To avoid performance issues, ensure that your searches are optimized if possible to return only the parts of objects that you need and not the full objects.

The following topics describe how contact search works between the core applications and ContactManager:

- “ContactManager support for contact searches” on page 92
- “ClaimCenter support for contact searches” on page 94
- “PolicyCenter support for Contact searches” on page 117
- “BillingCenter support for Contact searches” on page 118

ContactManager support for contact searches

ContactManager supports contact search that is initiated both locally, from its own user interface, and externally, through web services. External searches most commonly originate from Guidewire core applications.

Local ContactManager search criteria

If you run ContactManager and log in as a user who has permission to search for and view contacts, you can search for contacts by clicking the **Contacts** tab. The criteria you use for searches on this screen are defined in the ContactManager search-config.xml file and in the entity ABContactSearchCriteria.

IMPORTANT There are restrictions on adding new CriteriaDef elements. See the *Configuration Guide*.

See also

- For an example, see “Adding the InterpreterSpecialty property to search” on page 95.
- For complete information on search-config.xml, see the *Configuration Guide*

Defining minimum search criteria for a contact

There is a plugin class in the base configuration of ContactManager in which you can define the minimum criteria required to find a contact, gw.plugin.contact.ValidateABContactSearchCriteriaPluginImpl. Settings in this class affect both local searches and searches from Guidewire core applications. You can add new, required search fields by editing this class. Because the class is written in Gosu, you have considerable flexibility in how you specify which fields to require for a search.

The following code snippet from the parent class, ValidateABContactSearchCriteriaPluginBase, shows the default search criteria defined for an ABPerson entity, which you can override:

```
protected function abPersonCanSearch(searchCriteria : ABContactSearchCriteria) : boolean {
    if (searchCriteria.FirstName != null or searchCriteria.FirstNameKanji != null) {
        if (searchCriteria.Keyword == null and searchCriteria.KeywordKanji == null) {
            return false
        }
    }
    if (searchCriteria.Keyword == null
        and searchCriteria.KeywordKanji == null
        and searchCriteria.FirstName == null
        and searchCriteria.FirstNameKanji == null
        and searchCriteria.TaxID == null
        and satisfiesNoLocaleSpecificCriteriaRequirements(searchCriteria)
        and searchCriteria.Address.PostalCode == null
        and not searchCriteria.isValidProximitySearch()) {
        return false
    }
    return true
}
```

Effect of locale and country on minimum search criteria

Different locales have different requirements for searches. ContactManager determines the locale for the name fields of a search and uses the country specified to determine the other fields required.

The following code snippet from `ValidateABContactSearchCriteriaPluginBase.validateCanSearch` shows the code that picks up the locale and the country that the user specified in the search:

```
var country = searchCriteria.Address.Country ? gw.api.admin.BaseAdminUtil.getDefaultCountry()
if (searchSpec != null && searchSpec.Locale == null)
    searchSpec.Locale = LocaleType.get(PLConfigParameters.DefaultApplicationLocale.Value)
}
```

Note: If the country or the locale or both are not passed in, the method uses default values.

The `validateCanSearch` method throws an exception based on the contact subtype if the search criteria are not met:

```
var exception:ContactSubtypeSpecificException = null
if (searchCriteria.SpecialistServices != null and
    searchCriteria.SpecialistServices.Count > 0) {
    exception = new ContactSearchWithServiceCriteriaException(
        searchCriteria.ContactSubtypeType, country, searchSpec.Locale)
} else {
    exception = new TooLooseContactSearchCriteriaException(
        searchCriteria.ContactSubtypeType, country, searchSpec.Locale)
}
em = exception.Message
```

The message sent uses the locale and country to send the appropriate search criteria back to the calling application.

Note: In the class `ValidateABContactSearchCriteriaPluginImpl`, you can override the `satisfiesNoLocaleSpecificCriteriaRequirements` method to provide locale-specific search criteria. Your override can determine if the locale information can pass validation. For example, the base implementation requires a combination of City and State, but some locales might not use states or might have different requirements altogether.

See also

- “Minimum search criteria error messages” on page 93

Minimum search criteria error messages

The `ValidateABContactSearchCriteriaPluginImpl` class extends `ValidateABContactSearchCriteriaPluginBase`. This class uses the `TooLooseContactSearchCriteriaException` message definitions to pass locale and country specific error messages to the calling application. In the base configuration, these messages are display keys in the `display.properties` file.

To see this file, open Guidewire Studio for ContactManager. Navigate in the **Project** window to **configuration**→**config**→**Localizations**→**Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor.

In the file, if you press **Ctrl+F** and search for `TooLooseContactSearchCriteriaException`, you see the set of display keys defined for this exception. For example:

```
Java.TooLooseContactSearchCriteriaException.ABCompany.AU.ja_JP =
    Please specify one of the following\ : Name (phonetic), Name, Tax ID, Postcode, or City and State
```

The display keys are grouped first by entity type, `ABCompany`, `ABContact`, `ABPerson`, and `ABPlace`. In each set, there are display keys for the countries Australia (AU), Canada (CA), Germany (DE), France (FR), Great Britain (GB), Japan (JP), and the United States (US). For each country, there is a display key for one locale, `ja_JP`.

Note: You can add display keys for additional countries, locales, and entity types. If you add additional country or locale display keys, be sure to add them for all entity types, which in the base configuration are `ABCompany`, `ABContact`, `ABPerson`, and `ABPlace`.

This classification of messages enables ContactManager to return different contact search error messages based on country and locale. For example, a ClaimCenter user searches for a `MedicalCareOrg`, a `Company` subtype, located in

the United States. The user has previously set the locale to `ja_JP` (Japan). ContactManager determines that there are not enough search criteria to find a company in that country and locale, and sends an exception message defined by the following display key:

```
Java.TooLooseContactSearchCriteriaException.ABCompany.US.ja_JP =
Please specify one of the following\: Name (phonetic), Name, Tax ID, Zip Code, or City and State
```

Note: Each entity has one display key that does not specify a country or locale. This display key is a default in case the country or locale is not specified. Additionally, the display keys that have only a country and no locale are the default display keys for all locales that are not defined. In the base configuration, the only locale for which a message is defined is `ja_JP`.

ContactManager support for core application searches

The ContactManager web service `gw.webservice.ab.ab1000.abcontactapi.ABContactAPI` provides methods that the Guidewire core applications call to search for, create, retrieve, update, and delete contacts. Additionally, this class provides a method for finding vendor contacts by associated service. The method that supports search is `ABContactAPI.searchContact`. This class is read-only and cannot be directly edited.

You configure search in other classes and in a configuration parameter.

See also

- “ABContactAPI web service” on page 264

Configuring Contact search criteria and search results in ContactManager

The Gosu class `ABContactAPISearchCriteria` specifies the contact search criteria that the Guidewire core applications can use. The package for this class is `gw.webservice.ab.ab1000.abcontactapi`. You can edit this class to add new search criteria for core application contact searches.

You can also define the fields included in the search results that ContactManager sends back to a Guidewire core application. To add a field to the search results, add code for the field to the Gosu class

```
gw.webservice.ab.ab1000.abcontactapi.ABContactAPISearchResult.
```

These classes, `ABContactAPISearchCriteria` and `ABContactAPISearchResult`, are classes used as parameters in the web service API. After changing either of these classes, restart ContactManager to regenerate these web services. Then start Guidewire Studio™ for the Guidewire core application, navigate to the `ab1000.wsc` web collection, and fetch updates for the ContactManager web service `ABContactAPI`.

See also

- “Add search support in ContactManager for Guidewire core applications” on page 96

Limiting the number of service elements specified in a Contact search

If there are too many services specified in a contact search, the search can fail. To limit this number, there is a ContactManager configuration parameter, `MaxNumberServicesInSearchQuery`, that you can set in `config.xml`. This configuration parameter defaults to a value of 20. If a contact search specifies more services than the number set in this parameter, ContactManager sends an error message to the core application.

If `MaxNumberServicesInSearchQuery` is set to too large a number, ContactManager can experience SQL errors. In that case, ContactManager returns an error message to the core application with a note to check the logs in ContactManager for the details of the error.

ClaimCenter support for contact searches

There are two files that define the contact search criteria that ClaimCenter sends to ContactManager, the entity `ContactSearchCriteria.etx` and the Gosu class `ContactSearchMapper`. The entity defines the search criteria that appear in the search screens, and the Gosu class maps the search criteria to ContactManager. ClaimCenter uses these

criteria when it calls the ContactManager ABContactAPI method `searchContacts`. ClaimCenter calls this method in the plugin `ABContactSystemPlugin`, which implements `ContactSystemPlugin`.

There is a third Gosu class, `ContactSearchResultMapper`, that `ABContactSystemPlugin` uses to map fields sent from ContactManager search results so that ClaimCenter can display them in the lists of search results.

Note: ClaimCenter displays Contact entities in the search results. Therefore, the result mapper maps from ContactManager results to Contact entities.

The files are:

ContactSearchCriteria.etx

Defines the search criteria that are shown to the user in the search screens. If you add entries to `ContactSearchCriteria`, you must also add them to `ContactSearchMapper`. ClaimCenter uses both files.

gw.plugin.addressbook.ab1000.ContactSearchMapper

Defines how search criteria map to ContactManager ABContact search criteria.

gw.plugin.addressbook.ab1000.ContactSearchResultMapper

Defines the search results received from ContactManager to display in the search results screen.

See also

These files work in concert with the ContactManager search files described at “ContactManager support for core application searches” on page 94.

Adding the InterpreterSpecialty property to search

The entity `ABInterpreter` is a subtype of `ABPersonVendor` created in an example. It has one field, `InterpreterSpecialty`. Adding `InterpreterSpecialty` to search is a multi-step process that is split into a series of topics.

Before starting this process, complete the example “Adding a vendor contact subtype” on page 158.

IMPORTANT To improve search performance, Guidewire recommends that you add an index in the ContactManager entity definition for every field used in the search. The `ABInterpreter` example does define an index.

Add search support to the ContactManager user interface

Make changes to the ContactManager user interface to support searching for the `InterpreterSpecialty` field.

About this task


This step is the first in the multi-step process described at “Adding the InterpreterSpecialty property to search” on page 95.

In this step, you make changes that affect only the ContactManager user interface. You add `InterpreterSpecialty` to `ABContactSearchCriteria.etx`. Then you add a `CriteriaDef` element to the ContactManager `search-config.xml` file to specify where the `InterpreterSpecialty` column is located.

Procedure

1. In Guidewire Studio for ContactManager, in the **Project** window, navigate to **configuration**→**config**→**Extensions**→**Entity**, and then double-click `ABContactSearchCriteria.etx`.
2. In the editor, at the top of the **Element** hierarchy, click **nonPersistentEntity (extension)**.
3. Click the drop-down list next to **+** and choose **column**.
You use the **column** element rather than **typekey** because the data type is `varchar` and is not a `typekey`.
4. To the new column, add the following values that support searches for `InterpreterSpecialty`.

Name	Value
name	InterpreterSpecialty
type	varchar
nullok	true
desc	Interpreter language specialties

- Click the drop-down list next to  and choose **params**.
- Enter the following values to specify the size of this varchar column:

Name	Value
name	size
value	30

- In the **Project** window, navigate to **configuration→config→search** and double-click `search-config.xml` to edit the file.
- Increment the `version` attribute in the `SearchConfig` element at the top of the file.
In the following example, the original version number was 1 and has been changed to 2.

```
<SearchConfig
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="search-config.xsd"
  version="2">
```

- Find the following comment:

```
<!-- Specific fields in ABContact subtypes -->
```

- After the last `CriteriaDef` element in this group, add a new `CriteriaDef` element for the target entity `ABInterpreter` that tests `InterpreterSpecialty` for equality:

```
<CriteriaDef entity="ABContactSearchCriteria" targetEntity="ABInterpreter">
  <Criterion property="InterpreterSpecialty" matchType="eq"/>
</CriteriaDef>
```

Next steps

“Add search support in ContactManager for Guidewire core applications” on page 96

See also

- For a complete description of `search-config.xml`, see the *Configuration Guide*

Add search support in ContactManager for Guidewire core applications

Add code for the new `InterpreterSpecialty` field to the Gosu class `ABContactAPISearchCriteria`. This Gosu object is populated by the core application to pass a search criterion to `ContactManager`.

Before you begin

Complete “Add search support to the ContactManager user interface” on page 95.

About this task

For this example, the new field will be added to the search results that `ContactManager` sends back to the Guidewire core application. To add the field, you enter code for the field in the Gosu class `ABContactAPISearchResult`.

Procedure

1. Add an entry for `InterpreterSpecialty` to the class `ABContactAPISearchCriteria`.
 - a. In Guidewire Studio for ContactManager, navigate in the **Project** window to **configuration**→**gsrsrc** and then to `gw.webservice.ab.ab1000.abcontactapi`.
 - b. Double-click `ABContactAPISearchCriteria` to open the class in the editor.
 - c. Add the following variable definition to the list of variables at the beginning of the class.

```
public var InterpreterSpecialty : String
```

- d. In the method `toSearchCriteria`, before `:AllTagsRequired`, add the following entry:

```
:InterpreterSpecialty = this.InterpreterSpecialty,
```

2. If you want the field to be added to the search results sent back to a Guidewire core application, add an entry for `InterpreterSpecialty` to the class `ABContactAPISearchResult`.
 - a. Press `Ctrl+N` and enter `ABContactAPISearchResult`.
 - b. Click the class with package `gw.webservice.ab.ab1000.abcontactapi` in the search results to open the class in the editor.
 - c. Add the following variable definition to the list of variables at the beginning of the class.

```
public var InterpreterSpecialty : String
```

- d. In the constructor `construct(contact : ABContact)`, find the `if` statement block for `ABPerson`:

```
if (contact typeis ABPerson) {
```

- e. After that `if` statement block, add a new `if` statement for `ABInterpreter`, as follows:

```
// Search results for ABInterpreter
if (contact typeis ABInterpreter) {
    this.InterpreterSpecialty = contact.InterpreterSpecialty
}
```

Next steps

“Configure the address book search interface in ContactManager” on page 97

Configure the address book search interface in ContactManager

In this step of the search configuration example, you add the `InterpreterSpecialty` search field to the PCF files used in searching for address book contacts in ContactManager.

Before you begin

Complete “Add search support in ContactManager for Guidewire core applications” on page 96.

Procedure

1. If necessary, open Guidewire Studio for ContactManager.
2. Navigate in the **Project** window to **configuration**→**config**→**Localizations**→**Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor.
3. Press `CTRL+F` and enter `Web.ContactSearch.IncludePendingCreates` to find this entry.
4. Add a line and enter the following display key:
`Web.ContactSearch.InterpreterSpecialty = Interpreter Specialty`
5. Press `Ctrl+Shift+N` and enter `ContactSearchDV` to find this PCF file, and then double-click the file in the search results to open it in the editor.
6. Select the `ContactSearchDV` detail view panel.

- Click the **Code** tab at the bottom of the window and find the following line of code:

```
function isDoctor(c : ABContactSearchCriteria) : boolean {
    return entity.ABDoctor.Type.isAssignableFrom(c.ContactSubtypeType )}
```

- Insert a line after that one and enter on one line the following code:

```
function isInterpreter(c : ABContactSearchCriteria) : boolean {
    return entity.ABInterpreter.Type.isAssignableFrom(c.ContactSubtypeType )}
```

- In the **InputColumn** on the left, locate the **InputSet** containing the **Organization Name** input.
- From the **Toolbox** on the right, drag an **InputSet** widget and drop it under the **InputSet** containing the **Organization Name** input.
- Click the new **InputSet** and set the following property:

visible	isInterpreter(SearchCriteria)
----------------	-------------------------------

- Add an **Input** widget to the **InputSet** and set the following properties:

editable	true
id	InterpreterSpecialty
label	displaykey.Web.ContactSearch.InterpreterSpecialty
value	SearchCriteria.InterpreterSpecialty

Next steps

“Restart ContactManager and refresh web services in ClaimCenter” on page 98

Restart ContactManager and refresh web services in ClaimCenter

After you configure search for ContactManager, you must stop and restart ContactManager to force regeneration of the SOAP web services.

Before you begin

Complete “Configure the address book search interface in ContactManager” on page 97.

About this task

Before restarting ContactManager, you regenerate the ContactManager *Data Dictionary* to ensure that your changes to the entity **ABContactSearchCriteria** are valid. If they are, then you restart ContactManager and, in Guidewire Studio for ClaimCenter, you refresh the ContactManager web services.

Procedure

- Open a command prompt in the ContactManager installation folder and then enter the following command:

```
gwb stopServer
```

- Regenerate the dictionaries:

```
gwb genDataDictionary
```


- If the dictionary regenerates without errors, start the ContactManager application:

```
gwb runServer
```

- In Guidewire Studio for ClaimCenter, refresh the ContactManager plugin.
 - Press **Ctrl+Shift+N** and enter **ab1000.wsc** to find that web service collection, and then double-click the file in the search results to open it in the editor.

- b. In the editor, select the following resource:

```
${ab}/ws/gw/webService/ab/ab1000/abcontactapi/ABContactAPI?wsdl
```

- c. Click **Fetch**  to update the WSDL for this web service.

Next steps

“Add ContactManager search capability in ClaimCenter” on page 99

Add ContactManager search capability in ClaimCenter

Enable ClaimCenter to search for ContactManager contacts by InterpreterSpecialty.


Before you begin

Complete “Restart ContactManager and refresh web services in ClaimCenter” on page 98.


About this task

In this step you set search criteria and search results in ClaimCenter that work with ContactManager. The topic uses the new Contact subtype named Interpreter that has one field, InterpreterSpecialty.

Procedure

1. To support search on the InterpreterSpecialty field, add it to the entity ContactSearchCriteria.etx.
 - a. In Guidewire Studio for ContactManager, in the **Project** window, navigate to **configuration→config→Extensions→Entity**, and then double-click ContactSearchCriteria.etx.
 - b. In the editor, at the top of the **Element** hierarchy, select **nonPersistentEntity (extension)**.
 - c. Click the drop-down list next to  and choose **column**.
You use the **column** element because the data type is varchar and is not a typekey.
 - d. To the new column, add the following values that support searches for InterpreterSpecialty:

Name	Value
name	InterpreterSpecialty
type	varchar
nullok	true
desc	Interpreter language specialties

- e. Click the drop-down list next to  and choose **params**.
- f. Enter the following values to specify the size of this varchar column:

Name	Value
name	size
value	30

2. Add an entry for InterpreterSpecialty to the Gosu class ContactSearchMapper so you can use this field when you search for a contact.
 - a. In Guidewire Studio for ClaimCenter, navigate in the **Project** window to **configuration→gsrc** and then to **gw.plugin.contact.ab1000**.
 - b. Double-click ContactSearchMapper to open it in the editor.
 - c. Press CTRL+F and enter **convertToABContactAPISearchCriteria** to find this method.
 - d. Find the following line of code:

```
searchCriteriaInfo.FirstNameKanji = searchCriteria.FirstNameKanji
```

- e. After that line, add the following code:

```
searchCriteriaInfo.InterpreterSpecialty = searchCriteria.InterpreterSpecialty
```

3. Add an entry for `InterpreterSpecialty` to the Gosu class `ContactSearchResultMapper` so you can see this field in the search results that come back from `ContactManager`.
 - a. In the same **Project** window folder, `gw.plugin.contact.ab1000`, double-click the class `ContactSearchResultMapper` to open it in the editor.
 - b. Press `Ctrl+F` and enter `populateContactFromSearchResult` to find that method, and then in the method enter the following statement:

```
if (contact.typeis Interpreter) {
    contact.InterpreterSpecialty = searchResult.InterpreterSpecialty
}
```

Next steps

“Configure the address book search interface in ClaimCenter” on page 100

Configure the address book search interface in ClaimCenter

In this step of the search configuration example, you add the `InterpreterSpecialty` search field to the PCF files used in searching for Address Book contacts in ClaimCenter.

Before you begin

Complete “Add ContactManager search capability in ClaimCenter” on page 99.

Procedure

1. If necessary, open Guidewire Studio for ClaimCenter.
2. Navigate in the **Project** window to **configuration→config→Localizations→Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor.
3. Press `Ctrl+F` and enter the search text `Web.ContactSearch.IncludeSpecialistServices`, and then insert a new line after the one found by the search.
4. Enter the following display key on the new line:
`Web.ContactSearch.InterpreterSpecialty = Interpreter Specialty`
5. Press `Ctrl+Shift+N` and enter `AddressBookSearchDV` to find this PCF file, and then double-click the file in the search results to open it in the editor.
6. In the **InputColumn** on the left, locate the **InputSet** containing the **Law Firm Specialty TypeKeyInput**, and then copy the input set and paste the copy below it.
7. Select the new **InputSet** and set the following property:

visible	<code>searchCriteria.isSearchFor(entity.Interpreter)</code>
----------------	---

8. Right-click the **TypeKeyInput** widget inside the **InputSet** and click **Change element type**.
9. Click **Input** in the list of element types, and then click **OK**.
10. Set the following properties for the input widget:

editable	<code>true</code>
id	<code>InterpreterSpecialty</code>
label	<code>displaykey.Web.ContactSearch.InterpreterSpecialty</code>
value	<code>searchCriteria.InterpreterSpecialty</code>

Next steps

“Test the InterpreterSpecialty search extensions” on page 101

Test the InterpreterSpecialty search extensions

Before you begin

Complete “Configure the address book search interface in ClaimCenter” on page 100.

Procedure

1. If the ContactManager server is running, stop and then restart ContactManager.
2. Open a command prompt in the ClaimCenter installation folder and then enter the following command:

```
gwb stopServer
```

3. Regenerate the ClaimCenter *Data Dictionary* to test your changes to ContactSearchCriteria. At the command prompt, enter:

```
gwb genDataDictionary
```

4. If the data dictionary regenerates successfully, start the ClaimCenter application:

```
gwb runServer
```

5. Log in to ContactManager as a user who can create and edit contacts, such as user `aaplegate` with password `gw`.
6. Click **Search** on the left, and then choose **Interpreter** in the **Contact Type** list.
7. Search for one of the interpreters you created during your earlier tests.
 - a. Do a search for that interpreter and use the interpreter specialty as a search criterion to see if the search returns that interpreter.
 - b. Search for the same interpreter and use a specialty that is not a specialty for that interpreter to see if you get zero returns.
8. Log in to ClaimCenter and use the Address Book to search for an interpreter. Use the same search strategies as you did for ContactManager.

Adding an address field to Contact search

You can add an Address field to Contact search. The field can be either a normal address field or a denormalized address field.

Add a normal address field to search

You can add a normal address field, one that is not denormalized, to ClaimCenter search for ContactManager contacts.

About this task

The process for adding Address fields that are not denormalized fields to search is similar to the process for adding the InterpreterSpecialty property to search. However, the ClaimCenter part of the process is different enough that you might want to refer to the steps for adding a denormalized address field, starting with “Add a denormalized address field to search” on page 102.

See also

- “Adding the InterpreterSpecialty property to search” on page 95
- For the ContactManager part of the process, see “Adding the InterpreterSpecialty property to search” on page 95.

Add a denormalized address field to search

Adding a denormalized field to search can improve the speed of searching.

About this task

If you want to search on a denormalized address field, add fields to the `Address` entity and the `ABContact` entity to give `ContactManager` an actual field to search for. On the `ContactManager` side, the field you set up for search is the field you add to `ABContact`. On the core application side, the field you search for is the `Address` field, as it would be for a regular, non-denormalized `Address` field search. The following steps show what you do differently for denormalized `Address` searches:

Procedure

1. Add a column for the denormalized field to the `Address` entity and set `SupportsLinguisticSearch` to `true`.
2. Add a `searchColumn` to the `ABContact` entity that has the same name as the column added to `Address`, plus `Denorm`.

For example, if the `Address` column is named `County`, the `ABContact` column is named `CountyDenorm`. The `sourceColumn` is the same as the name for this search column, such as `CountyDenorm`, and the `sourceForeignKey` is `PrimaryAddress`.

3. The next steps are similar to those starting at “Add search support to the `ContactManager` user interface” on page 95. In general, for the `ContactManager` part of the process, use the `ABContact` search column as your field.

For example:

- a. Add a column to `ABContactSearchCriteria.etx`, using the name of the `ABContact` search column.
- b. Add to `search-config.xml` a `<Criterion>` for the column you just added to `ABContactSearchCriteria.etx`. Add it to the following `<CriteriaDef>` for `ABContact`:

```
<CriteriaDef entity="ABContactSearchCriteria" targetEntity="ABContact">
```

The new criterion must use the name of the column in `ABContactSearchCriteria.etx` and can have any match type you prefer.

4. The next steps are similar to those starting at “Add search support in `ContactManager` for Guidewire core applications” on page 96. In general, for the `ContactManager` part of the process, use the `ABContact` search column as your field.

For example:

- a. Add an entry for your `ABContact` search field to the class `ABContactAPISearchCriteria`.
 - b. If you want the field to be added to the search results sent back to a Guidewire core application, add the `ABContact` search field to the class `ABContactAPISearchResult`.
5. Refresh the web services. See “Restart `ContactManager` and refresh web services in `ClaimCenter`” on page 98.
 6. Go through the steps for the `ClaimCenter` part of the configuration, but use the `Address` field you want to have in search. This part of the configuration starts at “Add `ContactManager` search capability in `ClaimCenter`” on page 99.
 - a. The entity `ContactSearchCriteria.etx` has a foreign key to `Address`, so any field on `Address` can be used in contact search, and there is no need to update this file.
 - b. Add an entry for the `Address` field to `ContactSearchMapper` in the following `if` statement after the first line of code in that statement:

```
if (searchCriteria.Address != null) {
    var address = new ABContactAPIAddressSearch()
```

For example, for `Address.County`, add the following new line of code:

```
address.County = searchCriteria.Address.County
```

- c. The class `ContactSearchResultMapper` has a lot of the `Address` fields already in it. Check to see if a field you want to make visible for the address part of search might already be in the file. If you add a new field to `Address` that you want to search for, you must add an entry for the new `Address` field to this file.
- 7. Continue with the user interface configuration instructions, but use them as only a guideline for adding `Address` fields and do not follow them exactly. `Address` fields are located in different parts of the screens from `Contact` and `ABContact` fields. See “Configure the address book search interface in `ContactManager`” on page 97.

Adding the service state property to search

Adding a service state property to search is a multi-step process that is split into a series of topics. The service state property example uses the `ContactServiceState` entity and `ServiceState` property from the example in “Extending contacts with an array” on page 171.

In this example, you make the new `ServiceState` property searchable in both `ClaimCenter` and `ContactManager`. The property names must match in both applications. Any object you add to the search system must have indexes declared for its fields in `ContactManager` to make the search reasonably fast. In the example, `ContactServiceState` does have indexes declared for each of its fields. Without indexes, searching for the object can be slow.

Overview of adding search capability in `ContactManager`

Before starting the example of adding the `ServiceState` property to `ContactManager` and `ClaimCenter` search, be sure you understand how to use an `<ArrayCriterion>` element as described in this topic. Additionally, see “Adding the service state property to search” on page 103.

There are two primary aspects to adding search capability in `ContactManager`:

Adding search support to the `ContactManager` search screens

You add the `ServiceState` typekey to `ABContactSearchCriteria.etx`. Then you add an `<ArrayCriterion>` element to the `ContactManager` `search-config.xml` file to specify where the `ServiceState` column is located. There are restrictions on using `<ArrayCriterion>`:

- For any `<CriteriaDef>` element in the `search-config.xml` file, you must have only one `<ArrayCriterion>` subelement. If you have more than one `<ArrayCriterion>` defined for an entity, a search can return duplicate results for that entity. For example, you have two `<ArrayCriterion>` properties for `ABContact`. A successful search returns the same contact twice, once for each matched property.
- If you want to search for multiple fields in an array entity, create a new field that combines the fields.
- Additionally, the system requires unique values for the field that you are searching on. For example, `City` and `State` are two fields on an array entity, and the combination of the two fields is what makes the search unique. In this case, you must create a third field on the entity that concatenates the two values, and then create the array search criterion on that third field.

Adding search support for Guidewire core applications in the Gosu class `ABContactAPISearchCriteria`

You can have the search field show in the search results that `ContactManager` sends back to the Guidewire core application. To add the field to the search results, add code for the field to the Gosu class `ABContactAPISearchResult`.

These classes provide parameters used in web service calls, and changing them changes the web service WSDL. After changing one of these classes, you must restart `ContactManager` to regenerate the web APIs. Then, in Guidewire Studio for the Guidewire core application, you must refresh the `ContactManager` web APIs.

The first step in this example is “Enable `ServiceState` support for the `ContactManager` search screens” on page 103.

Enable `ServiceState` support for the `ContactManager` search screens

In the first step of this search example, you add support for the `ServiceState` property to enable its use in the `ContactManager` search screens.

Before you begin

Before you start this step, see the description of `<ArrayCriterion>` in “Overview of adding search capability in ContactManager” on page 103.

Procedure

1. In Guidewire Studio for ContactManager, in the **Project** window, navigate to **configuration→config→Extensions→Entity** and then double-click `ABContactSearchCriteria.etx` to open this entity in the editor.
2. Right-click **nonPersistentEntity (extension)** at the top of the **Element** hierarchy and click **Add new**, and then choose **typekey** from the drop-down list.
3. Enter the following values:

Name	Value
name	ServiceState
typelist	State
desc	State where contact provides services

4. In the **Project** window, navigate to **configuration→config→search** and double-click `search-config.xml` to edit the file.
5. Increment the version attribute in the `SearchConfig` element at the top of the file.
In the following example, the original version number was 1 and has been changed to 2.

```
<SearchConfig
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="search-config.xsd"
  version="2">
```

6. Press **Ctrl+F** and enter `ABContact` to find the `<CriteriaDef>` element for the target entity `ABContact`.
7. Add a new `<ArrayCriterion>` element to the end of this `<CriteriaDef>` element, as follows:

```
<!-- Search by ABContact Fields -->
<CriteriaDef entity="ABContactSearchCriteria" targetEntity="ABContact">
  <Criterion property="TaxID" matchType="eq"/>
  <Criterion property="VendorType" matchType="eq"/>
  <Criterion property="VendorAvailability" matchType="eq"/>
  <Criterion property="Keyword" matchType="startsWith"/>
  <Criterion property="KeywordKanji" matchType="startsWithCaseSensitive"/>
  <Criterion property="Score" matchType="ge"/>
  <!-- Some additional search criteria are defined in code -->
  <ArrayCriterion property="ServiceState"
    targetProperty="ContactServiceArea"
    arrayMemberProperty="ServiceState" />
</CriteriaDef>
```

The attributes of the new element are:

property

The name attribute for the column in the `ABContactSearchCriteria` entity

targetProperty

The name of the array on the base entity `ABContact`

arrayMemberProperty

The name of the column on the extension array entity `ContactServiceState`

Next steps

“Add `ServiceState` search support in ContactManager for core applications” on page 105

Add ServiceState search support in ContactManager for core applications

To support searching for contacts by the ServiceState property extension, add it to the ContactManager web API that core applications use for contact search.

Before you begin

Complete “Enable ServiceState support for the ContactManager search screens” on page 103. Also, it might be useful to review the information on the web API class ABContactAPISearchCriteria in “Overview of adding search capability in ContactManager” on page 103.

About this task

In this step of the search example, you add code for the ServiceState field to the Gosu class ABContactAPISearchCriteria. This web API class enables a Guidewire core application’s **Search** screen to show the field to the user and pass its value to ContactManager.

Procedure

1. In Guidewire Studio for ContactManager, navigate in the **Project** window to **configuration→gsrsrc** and then to **gw.webservice.ab.ab1000.abcontactapi**.
2. Double-click ABContactAPISearchCriteria to open it in the editor.
3. Add the following variable definition to the list of variables at the beginning of the class:
`public var ServiceState : typekey.State`
4. In the method toSearchCriteria, add a comma after the line starting with `:AllTagsRequired` and add a new entry for ServiceState:

```
:AllTagsRequired = this.AllTagsRequired,  
:ServiceState = this.ServiceState
```

Next steps

“Configure ContactManager search screens for ServiceState” on page 105

Configure ContactManager search screens for ServiceState

Enable ContactManager users to search for contacts by ServiceState extension.

Before you begin

Complete “Add ServiceState search support in ContactManager for core applications” on page 105.

About this task

Add the ServiceState search field to the PCF files used in searching for contacts in ContactManager.

Procedure

1. In Guidewire Studio for ContactManager, open the **Project** window.
2. Navigate in the **Project** window to **configuration→config→Localizations→Resource Bundle 'display'** and double-click **display.properties** to open this file in the editor.
3. Press Ctrl+F and enter the search text `Web.ContactSearch.Person.LastName` to find this line in the file, and then add a new line after it.
4. Add the following new entry:

```
Web.ContactSearch.ServiceState = Service State
```

5. In the **Project** window, navigate to **configuration→config→Page Configuration→pcf→search**, and then click **ContactSearchDV** to edit this PCF file.

6. In the InputColumn on the left, locate the InputSet containing the Organization Name input, then add an InputSet widget under it.
7. Click the new InputSet and set the following property:

visible	isCompany(SearchCriteria)
----------------	---------------------------

8. Add an Input widget to the InputSet and set the following properties:

editable	true
id	ServiceState
label	displaykey.Web.ContactSearch.ServiceState
value	SearchCriteria.ServiceState

Next steps

“Regenerate and refresh ContactManager web services” on page 106

Regenerate and refresh ContactManager web services

At this point in this multi-step search example, ClaimCenter does not have the web service changes you made in ContactManager. To get them, restart the ContactManager server to regenerate the web services, and then, in Guidewire Studio for ClaimCenter, refresh the ContactManager web service WSDL.

Before you begin

Complete “Configure ContactManager search screens for ServiceState” on page 105.

Procedure

1. Open a command prompt in the ContactManager installation folder and then enter the following command:

```
gwb stopServer
```

2. Start the ContactManager application:

```
gwb runServer
```

3. After ContactManager starts up, in Guidewire Studio for ClaimCenter, refresh the ContactManager plugin.
 - a. In the **Project** window, press Ctrl+Shift+N and enter ab1000.wsc. Then click ab1000.wsc in the search results to open this web service collection in the editor.
 - b. In the editor, select the following resource:

```
${ab}/ab/ws/gw/webservice/ab/ab1000/abcontactapi/ABContactAPI?wsdl
```

- c. Click **Fetch**  to update the WSDL.

Next steps

“Add ServiceState search capability in ClaimCenter” on page 106

Add ServiceState search capability in ClaimCenter

Before you begin

Complete “Regenerate and refresh ContactManager web services” on page 106.

Procedure

1. Using Guidewire Studio for ClaimCenter, in the **Project** window, navigate to **configuration→config→Extensions→Entity**.
2. Click `ContactSearchCriteria.etx` to open the entity extension in an editor.
3. Right-click **nonPersistentEntity (extension)** at the top of the **Element** hierarchy and click **Add new**, and then choose **typekey** from the drop-down list.
4. Enter the following values:

Name	Value
name	ServiceState
typelist	State
desc	State where contact provides services

5. Add an entry for `ServiceState` to the Gosu class `ContactSearchMapper` so you can use this field when you search for a contact.
 - a. In Guidewire Studio for ClaimCenter, press **Ctrl+N** and enter `ContactSearchMapper`, and then in the search results click `ContactSearchMapper(gw.plugin.contact.ab1000)` to open this class in the editor.
 - b. Press **Ctrl+F** and enter `convertToABContactAPISearchCriteria` to find this method in the class.
 - c. Insert a new line after the following line of code:

```
searchCriteriaInfo.PreferredVendors = searchCriteria.PreferredVendors
```

- d. Enter the following code for `ServiceState`:

```
searchCriteriaInfo.ServiceState = searchCriteria.ServiceState.Code
```

Next steps

“Configure ClaimCenter search screens for `ServiceState`” on page 107

Configure ClaimCenter search screens for `ServiceState`

Enable ClaimCenter users to search for contacts by `ServiceState` property extension.

Before you begin

Before you start this step, complete the step “Add `ServiceState` search capability in ClaimCenter” on page 106.

About this task

Add the `ServiceState` search field to the PCF files used in searching for contacts in ClaimCenter.

Procedure

1. If necessary, open Guidewire Studio for ClaimCenter.
2. Navigate in the **Project** window to **configuration→config→Localizations→Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor.
3. Press **Ctrl+F** and enter `Web.AddressBook.Search.SearchType` to find this line in the file, and then add a new line after it.
4. Add the following new entry:
`Web.AddressBook.Search.ServiceState = Service State`
5. In the **Project** window, navigate to **configuration→config→Page Configuration→pcf→addressbook** and double-click `AddressBookSearchDV` to edit this PCF file.
6. In the **InputColumn** on the left, locate the **InputSet** containing the Tax ID input, then add an **InputSet** widget under it.

7. Click the new InputSet and set the following property:

visible	<code>searchCriteria.isSearchFor(entity.Company)</code>
----------------	---

8. Add an Input widget to the InputSet and set the following properties:

editable	<code>true</code>
id	<code>ServiceState</code>
label	<code>displaykey.Web.AddressBook.Search.ServiceState</code>
value	<code>searchCriteria.ServiceState</code>

Next steps

“Test the ServiceState search extensions” on page 108

Test the ServiceState search extensions

Before you begin

Complete “Configure ClaimCenter search screens for ServiceState” on page 107.

Procedure

1. Shut down and restart both ClaimCenter and ContactManager.
 - a. Open a command prompt in the ContactManager installation folder and then enter the following command:


```
gwb stopServer
```
 - b. Start the ContactManager application:


```
gwb runServer
```
 - c. Open a command prompt in the ClaimCenter installation folder and then enter the following command:


```
gwb stopServer
```
 - d. Start the ClaimCenter application:


```
gwb runServer
```
2. Log in to ContactManager and click **Search** on the left, then choose **Company** in the **Contact Type** list.
3. Search for one of the companies you created during your earlier tests.
 - a. If you have not done so already, create a new company and specify a service state. Then do a search for that company with the **Service State** specified in the search to see if the search returns that company.
 - b. Search for the same company, first specifying a state that is not in the list of states for that company. See if you get zero returns.
 - c. Do a search that includes the first letter of that company name, but no state specified.
 - d. Do the same search with a state specified to see if the search results narrow to the companies with that value specified.
4. Log in to ClaimCenter and use the Address Book to search for a company that you know is stored in ContactManager. Use the same search strategies as you did for ContactManager.

Geocoding and proximity search for vendor contacts

ClaimCenter and ContactManager support *geocoding*, which enables the assignment of latitude and longitude to an address. By assigning latitude and longitude, the system is able to pinpoint an address as a location and specify its

geographic coordinates. The application can then use these geographic coordinates to present data like the distance between two addresses or all the addresses in a certain radius.

You can configure geocoding to work with ClaimCenter and ContactManager.

[See also](#)

Geocoding and batch processing

The geocoding feature uses the Guidewire work queue infrastructure for asynchronous searches. For example, ClaimCenter Geocode batch processing searches asynchronously for new addresses at the times specified in the `scheduler-config.xml` file, as described at “Schedule geocoding” on page 114.

ClaimCenter Geocode batch processing geocodes all addresses in the database that have not yet been geocoded. ContactManager AB Geocode batch processing does the same thing.

Configuring the geocoding work queues

The ClaimCenter `work-queue.xml` file configures one Geocode worker to geocode addresses.

```
<work-queue
  workQueueClass="com.guidewire.pl.domain.geodata.geocode.GeocodeWorkQueue"
  progressinterval="600000">
  <worker instances="1" batchsize="100"/>
</work-queue>
```

The ContactManager `work-queue.xml` file configures one ABGeocode worker to perform geocode processing on addresses.

```
<work-queue
  progressinterval="600000"
  workQueueClass="com.guidewire.ab.domain.geodata.geocode.ABGeocodeWorkQueue">
  <worker batchsize="100" instances="1"/>
</work-queue>
```

You can modify these configurations, but first verify that you have a good understanding of the work queue infrastructure. For example:

- You can have more than one instance of worker.
- One concept is how `progressinterval` is used with `batchsize` by the system. The default setting of `progressinterval` is 600,000 milliseconds, or 10 minutes. This setting is the amount of time that ClaimCenter or ContactManager allots for a worker to process `batchsize` geocode work items. If the time a worker has held a batch of items exceeds the `progressinterval`, the work items become *orphans*. ClaimCenter reassigns orphan work items to a new worker instance. Therefore, you need to set the `progressinterval` larger than the longest time required to process a work item, multiplied by the `batchsize`.

You must restart ClaimCenter and ContactManager after changing any of these files.

IMPORTANT If you have many new User or Contact objects in ClaimCenter or many new ABContact objects in ContactManager, processing these objects can be a system intensive operation. In this case, run the Geocode batch process when you anticipate that system use will be low, such as late at night.

[See also](#)

- For information on scheduling the batch process, see “Schedule geocoding” on page 114
- For information on work queues and the work queue scheduler, see the *System Administration Guide*

BatchGeocode and GeocodeStatus properties in ContactManager

AB Geocode batch processing in ContactManager determines which addresses to geocode by checking the BatchGeocode and GeocodeStatus properties. ContactManager handles these properties as follows:

- The Address property BatchGeocode indicates whether or not an address is to be geocoded. The default value is `false`, meaning that addresses by default are not geocoded.
- There is a ContactManager rule that sets vendor contacts to be automatically geocoded. The rule—Set Vendor's Primary Address BatchGeocode—is a Preupdate rule in the ABContactPreupdate rule set. This rule sets the BatchGeocode property of the primary address of an ABContact to `true` if the contact is tagged as a vendor.
- Geocode batch processing picks up addresses for which BatchGeocode is set to `true` and GeocodeStatus is set to `none`. This combination means that the address needs to be geocoded and has not been geocoded yet.

If you are adding many new contacts, especially into ContactManager, tune this parameter to match your expected daily load of new addresses.

See also

- “Schedule geocoding” on page 114

Run Geocode batch processing manually in ClaimCenter

About this task

You can run Geocode batch processing manually as a system administrator on the **Server Tools** screen.

Procedure

1. Log in as an administrator.
For example, enter username `su` and password `gw`.
2. Press `Alt+Shift+T` to access the **Server Tools** screen.
The screen opens with the **Server Tools** tab selected and the **Batch Process Info** item selected in the Sidebar.
3. Locate **Geocode Writer** and click its **Run** button.

Run Geocode batch processing manually in ContactManager

About this task

You can run Geocode batch processing manually as a system administrator on the **Server Tools** screen.

Procedure

1. Log in as an administrator.
For example, username `su` and password `gw`.
2. Press `Alt+Shift+T` to access the **Server Tools** tab.
3. In the sidebar, click **Batch Process Info**.
4. Locate **AB Geocode Writer** and click its **Run** button.

See also

- “Configuring the geocoding work queues” on page 109

Configuring geocoding for ClaimCenter and ContactManager

Configuring geocoding involves activating the geocoding plugin, setting `config.xml` parameters, and enabling the work queue and scheduler to run batch geocoding of addresses.

Configuring geocoding is a multi-step process.

IMPORTANT If your environment includes a ContactManager integration, you must perform these steps in both ClaimCenter and ContactManager for proper functioning of the geocoding feature.

1. “Activate geocoding in Bing Maps” on page 111
2. “Enable and use the Geocoding plugin” on page 111
3. “Set geocoding configuration parameters in ClaimCenter” on page 112
4. “Set geocoding configuration parameters in ContactManager” on page 113
5. “Schedule geocoding” on page 114
6. “Log geocoding information” on page 115
7. “Stop and restart ClaimCenter and ContactManager” on page 115

Activate geocoding in Bing Maps

To use the geocoding plugin, your company must set up its own account, login, and application key with Bing Maps.

Procedure

1. Go to <http://www.bingmapsportal.com>, where you can set up a Bing Maps account and obtain an application key.
When you create a key, the application name is arbitrary, and no application URL is required.
2. Register the geocoding plugin class that implements `GeocodePlugin` plugin interface.
In the base configuration, this plugin implementation class is `gw.plugin.geocode.impl.BingMapsPluginRest`.
Note: By default, the `GeocodePlugin` plugin interface uses the Microsoft Bing Maps web service and is disabled.

Next steps

“Enable and use the Geocoding plugin” on page 111

Enable and use the Geocoding plugin

For ClaimCenter and ContactManager to be able to use geocoding, you must enable the `GeocodePlugin` in both applications.

Before you begin

Complete “Activate geocoding in Bing Maps” on page 111.

About this task

The instructions for enabling the Geocoding plugin are the same for both ContactManager and ClaimCenter.

Procedure

1. Start Guidewire Studio™ for ClaimCenter and perform the steps that follow, and then start Guidewire Studio™ for ContactManager and perform the steps that follow.
2. In the **Project** window, navigate to **configuration→config→Plugins→registry** and then double-click `GeocodePlugin.gwp` to open this registry file in the editor.
3. Clear the **Disabled** check box to enable the plugin.
4. In the **Parameters** section, set the Geocode plugin parameters.

Parameter	Description
<code>applicationKey</code>	The application key that you obtained from Bing Maps. See “Activate geocoding in Bing Maps” on page 111.

Parameter	Description
geocodeDirectionsCulture	The locale for geocoded addresses and routing instructions returned from Bing Maps. For example, use the locale code ja-JP for addresses and instructions for Japan. The base application plugin implementation uses en-US if you do not specify a value. For a current list of codes that Bing Maps supports, see http://msdn.microsoft.com/en-us/library/cc981048.aspx .
imageryCulture	The language for map imagery. For example, use the language code ja for maps labeled in Japanese. The base application plugin implementation uses en if you do not specify a value. For a current list of codes that Bing Maps supports, see http://msdn.microsoft.com/en-us/library/cc981048.aspx .
mapURLHeight	Width of maps, in pixels. The base application plugin implementation uses 500 if you do not specify a value.
mapURLWidth	Height of maps, in pixels. The base application plugin implementation uses 500 if you do not specify a value.

Next steps

“Set geocoding configuration parameters in ClaimCenter” on page 112

Set geocoding configuration parameters in ClaimCenter

To enable geocoding features in the ClaimCenter user interface, you must set parameters in the `config.xml` file.

Before you begin

Complete “Enable and use the Geocoding plugin” on page 111.

About this task

If you have ContactManager integrated with ClaimCenter, you must consider both applications when you set some of these configuration parameters.

Procedure

1. Open Guidewire Studio™ for ClaimCenter.
2. In the **Project** window, navigate to **configuration→config**.
3. Double-click `config.xml` to open this file in an editor.
4. In the editor, press **Ctrl+F** and then enter **Geocoding** to find that section of the configuration parameters.
5. Set the following parameters to enable geocoding.

Parameter	Description
UseGeocodingInPrimaryApp	Set this parameter to true to enable geographical data and proximity search on application screens in ClaimCenter. The setting affects screens like Assignment and User search windows. This parameter must be true to perform assignment by proximity. The default setting is false.
UseGeocodingInAddressBook	Set this parameter to true to enable geocoding on ClaimCenter Address Book screens if ClaimCenter is integrated with ContactManager. The default setting is false.
UseMetricDistancesByDefault	Use kilometers in the user interface. The default setting, false, specifies that miles are to be used.

Parameter	Description
ProximitySearchOrdinalMaxDistance	<p>A distance that provides an approximate bound to improve performance of an ordinal (nearest <i>n</i>) proximity search. The search can return results that are farther away than the distance specified. The default setting is 300. The actual unit value of the distance is miles or kilometers depending on how you have set <code>UseMetricDistancesByDefault</code>.</p> <p>The setting for this parameter must be the same in <code>ContactManager</code> and <code>ClaimCenter</code>.</p> <p>This parameter has no effect on <i>radius</i> (within <i>n</i> miles or kilometers) proximity searches or walking-the-group-tree-based proximity assignment.</p>
ProximityRadiusSearchDefaultMaxResultCount	<p>Maximum number of results to return when performing a radius (<i>n</i> miles or kilometers) search from <code>ClaimCenter</code>.</p> <ul style="list-style-type: none">• <code>ClaimCenter</code> passes the value of this parameter to <code>ContactManager</code> when it makes a call for a proximity search.• The default value in the base configuration is 1000.• This parameter has no effect on ordinal (nearest <i>n</i>) proximity searches.• This parameter applies only to searches originating from <code>ClaimCenter</code> and does not have to match the value of the corresponding parameter in the <code>ContactManager config.xml</code> file.

Next steps

“Set geocoding configuration parameters in `ContactManager`” on page 113

Set geocoding configuration parameters in `ContactManager`

To enable geocoding features in the `ContactManager` user interface, you must set parameters in the `config.xml` file.

Before you begin

Complete “Set geocoding configuration parameters in `ClaimCenter`” on page 112.

Procedure

1. In Guidewire Studio for `ContactManager`, in the **Project** window, navigate to **configuration**→**config**.
2. Navigate to the bottom of that node, and then double-click `config.xml` to open the file in an editor.
3. In the editor, press **Ctrl+F** and then enter **Geocoding** to find that section of the configuration parameters.
4. Set the following parameters to enable geocoding.

Parameter	Description
UseGeocodingInAddressBook	<p>Set this parameter to <code>true</code> to enable geocoding for contacts storied in <code>ContactManager</code>. The default is <code>false</code>. The setting for this parameter must be the same in <code>ContactManager</code> and <code>ClaimCenter</code>.</p>

Parameter	Description
UseMetricDistancesByDefault	Use kilometers in the user interface. The default, false, is to use miles. The setting for this parameter must be the same in ContactManager and ClaimCenter.
ProximitySearchOrdinalMaxDistance	Maximum distance to use when performing an ordinal (nearest <i>n</i>) proximity search. The default is 300. The actual unit value of the distance is miles or kilometers depending on how you have set UseMetricDistancesByDefault. This parameter has no effect on radius (<i>n</i> miles or kilometers) searches or proximity assignment based on walking the group tree. The setting for this parameter must be the same in ContactManager and ClaimCenter.
ProximityRadiusSearchDefaultMaxResultCount	Maximum number of results to return when performing a radius (<i>n</i> miles or kilometers) search from the ContactManager search screen. <ul style="list-style-type: none"> The default value in the base configuration is 1000. This parameter has no effect on ordinal (nearest <i>n</i>) proximity searches. This parameter applies only to searches originating from ContactManager and does not have to match the value of the corresponding parameter in the ClaimCenter config.xml file.

Next steps

“Schedule geocoding” on page 114

Schedule geocoding

If you enable geocoding, you must also schedule the work queues that geocode addresses that have not been geocoded yet.

Before you begin

Complete “Set geocoding configuration parameters in ContactManager” on page 113.

About this task

When you schedule the geocoding work queue, use the following guidelines:

- Schedule Geocode and ABGeocode batch processing with enough time between runs to fully process the work items in the work queues. If you find duplicate work items in the work queues for the same address ID, make the time between runs a longer interval.
- Geocoding a large number of addresses can require considerable application resources. Set up your implementation to do its batch geocoding of addresses at a time when general access to your server is restricted.
- The ABGeocode batch process in ContactManager uses the settings described at “BatchGeocode and GeocodeStatus properties in ContactManager” on page 110.
- See also “Geocoding and batch processing” on page 109.

Procedure

- In Guidewire Studio for ClaimCenter, press Ctrl+Shift+N and enter scheduler-config.xml, and then double-click scheduler-config.xml in the search results to open it in the editor.
- In the editor, press CTRL+F and enter Geocode to find the following entry:

The following XML code shows the entry in the base configuration of ClaimCenter, including the `<!--` and `-->` comment tags:

```
<!-- New addresses searched for geocoding at 1:30 am -->
<!--
  <ProcessSchedule process="Geocode">
    <CronSchedule hours="1" minutes="30"/>
  </ProcessSchedule>
-->
```

This entry, when uncommented, instructs the batch process to start searching for new addresses to geocode every night at 1:30 AM.

3. Remove the opening `<!--` and closing `-->` comments to activate this entry the next time you start the application.
4. If ClaimCenter is integrated with ContactManager, you must also uncomment the ABGeocode entry in the ContactManager `scheduler-config.xml` file. In Guidewire Studio for ContactManager, follow the same procedure you did for ClaimCenter to open the file and find the entry.

The following XML code shows the entry in the base configuration of ContactManager, including the `<!--` and `-->` comment tags:

```
<!-- <ProcessSchedule process="ABGeocode">
      <CronSchedule minutes="0"/>
    </ProcessSchedule> -->
```

This entry, when uncommented, instructs the work queue to start searching for new addresses to geocode every hour on the hour.

5. Remove the opening `<!--` and closing `-->` comments to activate this entry the next time you start the application.

Next steps

“Log geocoding information” on page 115

Log geocoding information

Logging information for the geocoding plugins enables you to review results of the geocoding work queues.

Before you begin

Complete “Schedule geocoding” on page 114.

About this task

Perform the following tasks in both Guidewire Studio for ClaimCenter and Guidewire Studio for ContactManager.

Procedure

1. Navigate in the **Project** window to **configuration→config→logging** and then double-click `log4j2.xml`.
2. Configure the sections of this file that add details of geocoding plugin activity to the log.

Next steps

“Stop and restart ClaimCenter and ContactManager” on page 115

Stop and restart ClaimCenter and ContactManager

After making configuration and logging file changes for geocoding, you must stop the Guidewire applications that you configured if they are running, and then rebuild and redeploy them.

Before you begin

Complete “Log geocoding information” on page 115.

About this task

Following are the steps for stopping and starting both ClaimCenter and ContactManager when they are running in development mode:

Procedure

1. Stop ClaimCenter.

Open a command prompt in the ClaimCenter installation folder and then enter the following command:

```
gwb stopServer
```

2. Stop ContactManager.

Open a command prompt in the ContactManager installation folder and then enter the following command:

```
gwb stopServer
```

3. Start the ContactManager application in the ContactManager installation folder:

```
gwb runServer
```

4. Start the ClaimCenter application in the ClaimCenter installation folder:

```
gwb runServer
```

See also

- For information on geocoding plugin integration, see the *Integration Guide*
- For information on configuring geocoding, see the *System Administration Guide*

Proximity search example for vendor contacts

Your search results are returned from ContactManager ordered by distance, closest to farthest away. In the base configuration, this order of search results is the only one available.

The system sorts the results of a proximity search, but it does not display driving directions with the search results. To obtain driving directions, you must select a return address and click the **Return Driving Directions** button. Then additional columns for driving distances and times appear, as well as links to display the directions for the requested search results.

Note: You cannot sort driving direction results.

Request distanced-based proximity search

This example shows you how to request a distance-based proximity search from the ClaimCenter user interface and describes what the system does in response to the search request.

Before you begin

To walk through this example in ClaimCenter, you must have done the following:

- Integrated ContactManager with ClaimCenter as described in “Integrating ContactManager with Guidewire core applications” on page 59.
- Loaded sample data as described at “Load sample data for ContactManager” on page 56.
- Set up geocoding for both applications as described in “Configuring geocoding for ClaimCenter and ContactManager” on page 110.
- Run Geocode batch processing for ClaimCenter and ABGeocode batch processing for ContactManager as described in:
 - “Run Geocode batch processing manually in ClaimCenter” on page 110
 - “Run Geocode batch processing manually in ContactManager” on page 110

About this task

In this example, you request the closest auto repair shops within 25 miles of San Mateo, California, USA.

Procedure

1. Log in to ClaimCenter.
2. Click the **Address Book** tab.
3. On the **Search Address Book** screen, choose the following settings:

Type	Auto Repair Shop
Search Radius	25 miles
City (under Search Radius)	San Mateo
ZIP Code	94404

4. Click **Search**.
5. ClaimCenter determines the center of the proximity search by synchronously geocoding this one address.
6. The system applies any filters that you set in the search. For example, if under **Location** you set the **City** to Burlingame and the **State** to California, the search eliminates every address that does not have a city set to Burlingame. Any address without a latitude and longitude is not considered.
7. ClaimCenter calculates the distance of the remaining results from the search center. If you limited the range as shown previously, ClaimCenter discards any results that are too far away.
8. ClaimCenter sorts the remaining results in ascending order of proximity, from closest to farthest away, and returns them in the **Search Results**.

PolicyCenter support for Contact searches

PolicyCenter enables you to search both for locally stored contacts and for contacts stored externally in ContactManager.

To configure Contact search in PolicyCenter, you edit the following files in Guidewire Studio for PolicyCenter:

- `search-config.xml` – Defines search criteria for addresses.
- `gw.plugin.contact.ab1000.ABContactSystemPlugin` – The plugin implementation used to call ContactManager web services. This plugin implementation has a `searchContacts` method that calls the ContactManager `ABContactAPI.searchContact` method.
- `gw.plugin.contact.ab1000.ABContactAPISearchCriteriaEnhancement` – The class that defines additional PolicyCenter search criteria to be used with ContactManager contact searches. The class enhances the

ContactManager web service `ABContactAPISearchCriteria`, adding a `sync` method that is used by `ABContactSystemPlugin.searchContacts`.

- `gw.plugin.contact.ab1000.ABContactAPISearchResultEnhancement` – The class that defines the search result fields that PolicyCenter displays in the search results returned from ContactManager.
- `ContactSearchCriteria.etx` – You can extend this entity by creating a `ContactSearchCriteria.etx` file. This entity defines the search criteria that are shown to the user in the search screens. If you add search criteria to `ContactSearchCriteria.etx`, you must also add them to `ABContactAPISearchCriteriaEnhancement`. PolicyCenter uses both files.
- `gw.plugin.contact.impl.ContactSearchCriteriaEnhancement` – The class in which the contact search method for both internal and external contact search is defined. The primary contact search method in this class is `performSearch`.
- `ContactSearchScreen.pcf` – This PCF file displays the contact search criteria. It also instantiates an instance of the `ContactSearchCriteria` entity and uses it as a parameter to its `doSearch` method defined in the **Code** tab. That method calls the method `ContactSearchCriteriaEnhancement.performSearch`.

In the base configuration, PolicyCenter uses the Address fields defined in `search-config.xml` to search only for address fields. PolicyCenter does not use this file to configure searching for Contact fields.

To search for contacts stored in ContactManager, PolicyCenter uses the plugin implementation `gw.plugin.contact.ab1000.ABContactSystemPlugin`. This class's `searchContacts` method instantiates an `ABContactAPISearchCriteria` object and calls its `sync` method, which in turn calls the ContactManager web service method `ABContactAPI.searchContact`. The PolicyCenter `searchContacts` method returns data from the `ABContactAPISearchResult` Gosu objects returned by the ContactManager `searchContact` method. PolicyCenter shows these returned objects as contacts and their fields in the search results.

One class where you can customize this contact search behavior is

`gw.plugin.contact.ab1000.ABContactAPISearchCriteriaEnhancement`. This class extends the search criteria defined in the ContactManager web service `ABContactAPISearchCriteria`. For example, the `sync` method specifically searches for the client tag. If you want to search for other tags, you can modify the code of this method to do so.

A corresponding class in which you can customize contact search behavior is

`gw.plugin.contact.ab1000.ABContactAPISearchResultEnhancement`. This class defines extensions to the search results returned by ContactManager. It enhances the ContactManager web service `ABContactAPISearchResult`.

See also

- For information on the ContactManager files involved in an external contact search, see “ContactManager support for core application searches” on page 94.

BillingCenter support for Contact searches

BillingCenter enables you to search both for locally stored contacts and for contacts stored externally in ContactManager.

To configure Contact search in BillingCenter, you edit the following files in Guidewire Studio for BillingCenter:

- `search-config.xml` – Defines search criteria for addresses.
- `gw.plugin.contact.ab1000.ABContactSystemPlugin` – The plugin implementation used to call ContactManager web services. This plugin implementation has a `searchContacts` method that calls the ContactManager `ABContactAPI.searchContact` method.
- `gw.plugin.contact.ab1000.ABContactSearchCriteriaInfoEnhancement` – The class that defines additional BillingCenter search criteria to be used with ContactManager contact searches. The class enhances the

ContactManager web service `ABContactAPISearchCriteria`, adding a `sync` method that is used by `ABContactSystemPlugin.searchContacts`.

- `gw.plugin.contact.ab1000.ContactResultFromSearch` – The class that defines the search result fields that BillingCenter displays in the search results returned from ContactManager.
- `ContactSearchCriteria.eix` – You can extend this entity by creating a `ContactSearchCriteria.etx` file. `ContactSearchCriteria` defines the search criteria that are shown to the user in the search screens. If you add search criteria to `ContactSearchCriteria.etx`, you must also add them to `ABContactSearchCriteriaInfoEnhancement`. BillingCenter uses both files.
- `gw.plugin.contact.impl.ContactSearchCriteriaEnhancement` – The class in which the contact search method for both internal and external contact search is defined. The primary contact search method in this class is `performSearch`.
- `ContactSearchScreen.pcf` – This PCF file contains the display view in which the contact search criteria are shown. `ContactSearchScreen.pcf` also instantiates an instance of the `ContactSearchCriteria` entity and uses it as a parameter to its `doSearch` method defined in the **Code** tab. That method calls the method `ContactSearchCriteriaEnhancement.performSearch`.
- `ContactSearchDV.pcf` – This PCF file contains the contact search criteria shown in the search screen. This file is where you add new search criteria.

In the base configuration, BillingCenter uses the `Address` fields defined in `search-config.xml` to search for address fields. However, to maintain performance, any search that uses the `Address` field must also include at least one of the following fields: `Company Name`, `First Name`, or `Last Name`. These minimally-required search criteria are defined in the `isReasonablyConstrainedForSearch` method in the `ContactCriteria` class.

To search for contacts stored in ContactManager, BillingCenter uses the plugin implementation `gw.plugin.contact.ab1000.ABContactSystemPlugin`. This class's `searchContacts` method instantiates an `ABContactAPISearchCriteria` object and calls its `sync` method, which in turn calls the ContactManager web service `ABContactAPI.searchContact`. The BillingCenter `searchContacts` method returns data from the `ABContactAPISearchResult` Gosu objects returned by the ContactManager `searchContact` method. BillingCenter shows these returned objects as contacts with their fields in the search results.

One class in which you can customize this contact search behavior is `gw.plugin.contact.ab1000.ABContactSearchCriteriaInfoEnhancement`. This class enhances the search criteria defined in the ContactManager web service `ABContactAPISearchCriteria`. For example, the `sync` method of this class specifically searches for the `Client` tag. If you want to search for other tags, you can modify the code of this method to do so.

A corresponding class where you can customize contact search behavior is `gw.plugin.contact.ab1000.ContactResultFromSearch`. This class defines extensions to the search results returned by ContactManager. It extends the ContactManager web service `ABContactAPISearchResult`.

See also

- For information on the ContactManager files involved in an external contact search, see “ContactManager support for core application searches” on page 94.

Securing access to contact information

You can grant secure access to the information associated with contacts, both in ContactManager and in the core applications.

A fundamental component of enforcing security on shared contacts is checking permissions for users of the Guidewire core application. Contact security can be enforced both by a core application and by ContactManager as users create, update, and delete contacts. Additionally, users of ContactManager with appropriate permissions can merge duplicate contacts and review pending contact updates and creation.

Each Guidewire application has its own rules and security setup.

See also

- “ContactManager contact security” on page 121
- “PolicyCenter contact security” on page 132
- “BillingCenter contact security” on page 132
- “Security” in the *BillingCenter Application Guide*
- “ClaimCenter contact security” on page 134
- “Configuring ClaimCenter contact security” on page 137
- “Security: Roles, Permissions, and Access Controls” in the *ClaimCenter Application Guide*

ContactManager contact security

ContactManager is built on the same platform as the Guidewire core applications and provides roles and permissions that you can configure to control access to contact data. Additionally, the permissions have a set of permission check expressions you can configure to limit access to specific screens and widgets.

Role-based security defines what actions a user of a Guidewire application is allowed to perform. This type of security includes defining permissions, grouping related permissions into roles, and assigning these roles to users based on the work they perform in the Guidewire application.

For most purposes, other than merging duplicate contacts and reviewing pending contact changes, you typically access ContactManager data through a core application, like ClaimCenter, PolicyCenter, or BillingCenter. Those applications provide contact security to control which users of the application can access contact data.

You can set up ContactManager users to access contact data directly in ContactManager. The base configuration provides a role for this purpose, the ContactManager role.

The primary tasks that users have to perform in ContactManager are:

Merging duplicate contacts

See “Detecting and merging duplicate contacts” on page 247.

Reviewing pending contact changes

See “Review pending changes to contacts” on page 253.

See also

- “ContactManager user roles” on page 122
- “PolicyCenter contact security” on page 132
- “BillingCenter contact security” on page 132
- “ClaimCenter contact security” on page 134

ContactManager user roles

A role is a collection of permissions. By grouping permissions into roles, you can define the authority of a user of ContactManager by assigning the user a few roles rather than a larger list of permissions. A user can have multiple roles and must have at least one role.

The permissions used for contact access in the base configuration of ContactManager are `abcreate`, `abcreatepref`, `abdelete`, `abdeletepref`, `abedit`, `abeditpref`, `abview`, `abviewpending`, `abviewmerge`, `abviewpending`, `abviewsearch`, `anytagcreate`, `anytagdelete`, `anytagedit`, and `anytagview`.

These permissions are described in more detail in the topic, “ContactManager contact subtype and tag permissions” on page 124.

You can create additional permissions for certain contacts, contact subtypes, or tags. For example, you can create a permission for working with client contacts and another for working with vendor contacts.

To add permissions to roles and assign roles to users, use the ContactManager **Roles** screen. Log in as a user with administration privileges and click the **Administration** tab, and then navigate in the sidebar to **Users & Security**→**Roles**. See “Configuring ContactManager contact security” on page 127.

The base configuration of ContactManager provides the following roles, each of which has a set of default permissions. It is likely that you will add your own roles and permissions as well.

Role	Permissions	Description
Client Application	<ul style="list-style-type: none"> • Client Application – <code>clientapp</code> • Create address book contacts – <code>abcreate</code> • Create address book preferred vendors – <code>abcreatepref</code> • Add documents to a contact – <code>doccreate</code> • Create contact with any tag – <code>anytagcreate</code> • Delete address book contacts – <code>abdelete</code> • Delete address book preferred vendors – <code>abdeletepref</code> • Delete contact with any tag – <code>anytagdelete</code> • Edit address book contacts – <code>abedit</code> • Edit address book preferred vendors – <code>abeditpref</code> • Edit contact with any tag – <code>anytagedit</code> • Edit documents – <code>docedit</code> • Edit user language – <code>usereditlang</code> • Remove documents from a contact – <code>docdelete</code> • View address book contact search screens – <code>abviewsearch</code> • View address book contacts – <code>abview</code> • View contact with any tag – <code>anytagview</code> • View documents – <code>docview</code> 	This role is used by core applications to communicate with ContactManager. It contains permissions that allow the core applications to perform specific tasks.

Role	Permissions	Description
ContactManager	<ul style="list-style-type: none"> Create address book contacts – abcreate Create address book preferred vendors – abcreatepref Create contact with any tag – anytagcreate Add documents to a contact – doccreate Delete address book contacts – abdelete Delete address book preferred vendors – abdeletepref Delete contact with any tag – anytagdelete Edit address book contacts – abedit Edit address book preferred vendors – abeditpref Edit contact with any tag – anytagedit Edit documents – docedit Edit user language – usereditlang View address book contact search screens – abviewsearch View address book contacts – abview View contact with any tag – anytagview View documents – docview View merge – abviewmerge View pending – abviewpending 	User with full permission to create, edit, and delete contacts.
Contact Subtype Changer	<ul style="list-style-type: none"> Change Contact Subtype – changecontactsubtype SOAP administration – soapadmin 	User with permissions to change the subtype of a contact instance.
Contact Viewer	<ul style="list-style-type: none"> Edit user language – usereditlang View address book contact search screens – abviewsearch View address book contacts – abview View contact with any tag – anytagview 	User with view-only permissions for contacts
Data Protection Officer	<ul style="list-style-type: none"> Create groups – groupcreate Delete groups – groupdelete Edit groups – groupedit Edit obfuscated user contact – editobfuscatedusercontact Edit user language – usereditlang Edit users – useredit Grant roles to users – usergrantroles Request Contact Destruction – requestcontactdestruction View all users – userviewall View group tree – grouptreeview View groups – groupview View user – userview 	User who can respond to failures in contact purging or obfuscation with corrections
Rule Admin	<ul style="list-style-type: none"> Administer rules – ruleadmin Edit user language – usereditlang 	Rule administrator
Tools View	<ul style="list-style-type: none"> View BatchProcess tools screen – toolsBatchProcessview View Cache Info screen – toolsCacheinfoview View Cluster tools screen – toolsClusterview View Info tools screen – toolsInfoview View Log tools screen – toolsLogview View ManagementBeans tools screen – toolsJMXBeansview View Profiler tools screen – toolsProfilerview View StartablePlugin tools screen – toolsPluginview View WorkQueue tools screen – toolsWorkQueueview 	<p>User with permission to work on the Server Tools screens.</p> <p>To access Server Tools, press Alt+Shift+T and click Server Tools.</p>

Role	Permissions	Description
User Admin	<ul style="list-style-type: none"> • Always access debug tools – usereditattrs • Create groups – groupcreate • Create users – usercreate • Delete groups – groupdelete • Delete users – userdelete • Edit groups – groupedit • Edit user language – usereditlang • Edit users – useredit • Grant roles to users – usergrantroles • Manage attributes – attrmanage • Manage holidays – holidaymanage • Manage regions – regionmanage • Manage roles – rolemanage • Manage script parameters – scrprmmmanage • Manage security zones – seczonemmanage • Resync message – resyncmessage • Retry message – retrymessage • Skip message – skipmessage • SOAP administration – soapadmin • View attributes – attrview • View event messages – eventmessageview • View group tree – grouptreeview • View groups – groupview • View holidays – holidayview • View regions – regionview • View roles – roleview • View script parameters – scrprmvview • View user – userview 	User who handles administration of ContactManager users.

ContactManager contact subtype and tag permissions

The Guidewire core applications and ContactManager provide contact subtype and tag permissions that you can use to control access to contacts. The SystemPermissionType typelist lists all the subtype and tag permissions in ContactManager.

The following table lists the subtype and tag permissions provided in the base configuration of ContactManager for contacts:

Code	Permission Description
abview	View the details of contact entries in ContactManager
abviewsearch	View ContactManager contact search screens
anytagcreate	Create a new contact regardless of which contact tag it requires
anytagdelete	Delete a contact that has any contact tag
anytagedit	Edit a contact that has any contact tag
anytagview	See a contact that has any contact tag
abcreate	Create a new contact in ContactManager
abcreatepref	Create a new preferred vendor in ContactManager
abdelete	Delete an existing contact from the address book
abdeletepref	Delete an existing preferred vendor address book entry

Code	Permission Description
abedit	Edit an existing contact in ContactManager
abeditpref	Edit an existing preferred vendor in ContactManager
abviewmerge	Review and merge duplicate contacts in the Merge Contacts screens
abviewpending	Review and approve or disapprove pending contacts in the Pending Changes screens

The system uses role-based security for these permissions. As described in the previous topic, to implement role-based security, a system administrator associates permissions with roles in the system and assigns roles to users. For each role assigned, the user acquires the permissions associated with that role. For example, a role associated with the `abcreate` and `anytagcreate` permissions enables the user who has this role to create any type of contact.

The contact and tag permissions supplied in the base configuration apply across all contact subtypes and tags. If you create a permission that applies to a contact subtype, that permission also applies to all the subtypes of that contact subtype.

ContactManager enables you to restrict permissions according to contact subtype or tag. For example, you can enable a user with `abcreate` permission to create only `PersonVendor` contacts, but not `CompanyVendor` contacts. You configure contact and tag permissions through the `SystemPermissionType` typelist and the `security-config.xml` resource.

Note: If you create a set of tag permissions for a specific tag, these permissions enable access to contacts that have only that tag. For example you create a set of `Vendor` tag permissions and a user has a role with only those tag permissions. That user will not be able to work with a contact that has both `Claim Party` and `Vendor` tags. You could also create a set of tag permissions for `Claim Party` tags. In that case, a user with both `Vendor` and `Claim Party` tag permissions would be able to work with contacts that have both `Vendor` and `Claim Party` tags.

For examples showing how to create and use permissions for a contact subtype and a contact tag, see “Configuring ContactManager contact security” on page 127.

ContactManager contact and tag permission check expressions

In a page configuration file (PCF), you can control permissions on specific widgets with Gosu expressions that determine if a user has permission to perform an operation.

For example, in the `ContactManager NewContact.pcf` file, the `canVisit` attribute is set to:

```
perm.ABContact.create(ContactType) and
ContactTagType.userHasCreatePermissionForAtLeastOneContactTagType()
```

These expressions control access to this page. They allow access only to users who have both subtype permission to create the contact and at least one tag permission to create a tag.

Note: To see this PCF file, in Guidewire Studio™ for ContactManager, press `Ctrl+Shift+N` and enter `NewContact`, and then double-click `NewContact.pcf` (`configuration\config\web\pcf\contacts`) in the list of objects that the system finds.

Permission check expression parameters in ContactManager

Some Gosu permission check expressions require a parameter, and some do not.

For example, the `ContactManager` tag create permission check expression is the same as the `ABContact` create permission check expression: `perm.ABContact.create`. When this expression has a `ContactTagType` argument, the expression verifies that the user has permission to create a contact with this tag. The `ContactTagType` arguments for this expression in the base configuration support the following enumeration constants:

- `ContactTagType.TC_CLIENT`
- `ContactTagType.TC_VENDOR`
- `ContactTagType.TC_CLAIMPARTY`

For example, the expression `perm.ABContact.create(ContactTagType.TC_CLIENT)` verifies that the user has permission to create a new contact with the Client tag. If the user has `anytagcreate` permission, which grants permission for creating contacts with all contact tag types, this permission check returns `true`.

The contact and tag create permission expression can take an `ABContact` type or subtype parameter or a `ContactTagType` typecode specified as an enumeration constant. The contact and tag delete, edit, and view permission expressions take an `ABContact` instance as a parameter. All these expressions check for both contact and tag permissions, so no additional tag permission check expression is needed. These expressions are:

- `perm.ABContact.create`
- `perm.ABContact.delete`
- `perm.ABContact.edit`
- `perm.ABContact.view`

The following table lists the ContactManager contact and tag permission check expressions:

ContactManager Expression	Description
<code>perm.ABContact.create</code>	Takes an input parameter that is either an <code>ABContact</code> type or subtype or a <code>ContactTagType</code> typecode specified as an enumeration constant. Depending on the parameter, verifies that the user has the permission to create either a contact with the specified tag or a contact of the specified type.
<code>perm.ABContact.createpreferred</code>	Does not take an input parameter. Verifies that the user has permission to add the preferred contact in the address book.
<code>perm.ABContact.delete</code>	Takes an <code>ABContact</code> instance as an input parameter. Determines the subtype and contact tag or tags from the instance, and then verifies that the user has the contact and tag permissions to delete the contact.
<code>perm.ABContact.deletepreferred</code>	Does not take an input parameter. Verifies that the user has permission to delete the preferred vendor from the address book.
<code>perm.ABContact.edit</code>	Takes an <code>ABContact</code> instance as an input parameter. Determines the subtype and contact tag or tags from the instance, and then verifies that the user has the contact and tag permissions to edit the contact.
<code>perm.ABContact.editpreferred</code>	Does not take an input parameter. Verifies that the user has permission to edit the preferred vendor in the address book.
<code>perm.ABContact.view</code>	Takes an <code>ABContact</code> instance as an input parameter. Determines the subtype and contact tag or tags from the instance, and then verifies that the user has the contact and tag permissions to view the contact.
<code>perm.ABContact.viewmerge</code>	Does not take an input parameter. Verifies that the user has permission to merge contacts in the address book.
<code>perm.ABContact.viewpending</code>	Does not take an input parameter. Verifies that the user has permission to view pending contacts in the address book.
<code>perm.ABContact.viewsearch</code>	Does not take an input parameter. Verifies that the user has permission to search for contacts in the address book.

Viewing permissions in the ContactManager Security Dictionary

You can use the *Security Dictionary* to get information on the application permission keys. For example, after opening the *Security Dictionary*, click the **System Permissions** filter at the top of the left pane. You see all the permissions listed on the left by code name. If you click the permission code `abedit`, you see that it has the related application permission key **ABContact edit**, which has the associated Gosu check expression `perm.ABContact.edit`. You can filter the list by application permission keys, pages, system permissions, and roles.

Using the *Security Dictionary*, you can determine the following:

- The system permission related to an application permission key
- PCF files and widgets that use an application permission key
- Roles, application permission keys, PCF pages, and widgets that use a system permission
- Gosu application permission expressions called from each PCF page
- Permissions assigned to each role

See also

- “Build and view the ContactManager Security Dictionary” on page 127

Build and view the ContactManager Security Dictionary

Procedure

1. At a command prompt, run the following command from the ContactManager installation folder:

```
gwb genDataDictionary
```

This command builds the *Security Dictionary* in the following location:

ContactManager/build/dictionary/security/index.html

2. To see the *Security Dictionary*, navigate in a web browser to the location of the dictionary. For example, open:
file:///C:/ContactManager/build/dictionary/security/index.html

Contact search security configuration parameters in ContactManager

There are two parameters you can set in the ContactManager config.xml file to configure security for contact searches, RestrictSearchesToPermittedItems and RestrictContactPotentialMatchToPermittedItems. You edit this file in Guidewire Studio for ContactManager.

You can use the RestrictSearchesToPermittedItems configuration parameter to control the interaction between the permissions abviewsearch and abview. The abviewsearch permission determines which users can search for contacts. However, not all users with abviewsearch permission also have abview permission. The abview permission enables users to view the contact’s detailed information.

If RestrictSearchesToPermittedItems is false, in response to a search the system returns all contacts that match the search criteria. If this parameter is true, the system returns only contacts for which the user has view permissions. This setting also interacts with contact and tag permissions. For example, if a user can view only the Person subtype with any tag and RestrictSearchesToPermittedItems is true, the system returns only contacts of the Person subtype.

Additionally, you can set the RestrictContactPotentialMatchToPermittedItems parameter. This parameter controls the security of potential search results. If the parameter is true, only the potential matches for which the user has view permissions are returned.

Configuring ContactManager contact security

Use the SystemPermissionType typelist and the security-config.xml configuration file to define new ContactManager contact security permissions. These resources enable you to create more finely grained system permissions for specific Contact subtypes and tags.

Create permissions for an ABContact subtype in ContactManager

Procedure

1. If necessary, start Guidewire Studio for ContactManager.

At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb studio
```

2. In Guidewire Studio for ContactManager, press Ctrl+Shift+N and search for `SystemPermissionType.ttx`.
3. Double-click `SystemPermissionType.ttx` in the search results to open this typelist in an editor.
4. Add one or more new typecodes to `SystemPermissionType.ttx`.

For example, for each of the following contact permission typecodes, right-click an existing typecode and choose **Add new**→**typecode**. Then enter the information for the new typecode.

Code	Name	Description
abpersoncreate	Create an ABPerson instance	Permission to create an instance of an ABPerson subtype
abpersonview	View an ABPerson instance	Permission to view an instance of an ABPerson subtype
abpersonedit	Edit an ABPerson instance	Permission to edit an instance of an ABPerson subtype
abpersondelete	Delete an ABPerson instance	Permission to delete an instance of an ABPerson subtype

5. In the **Project** window, navigate to **configuration**→**config**→**security** and then double-click `security-config.xml`.
6. Associate the new permissions with an ABContact subtype in the `security-config.xml` file.

For example, map the new typecodes to the ABPerson contact subtype as follows:

```
<ContactPermissions>
  <ContactSubtypeAccessProfile entity="ABPerson">
    <ContactCreatePermission permission="abpersoncreate"/>
    <ContactViewPermission permission="abpersonview"/>
    <ContactEditPermission permission="abpersonedit"/>
    <ContactDeletePermission permission="abpersondelete"/>
  </ContactSubtypeAccessProfile>
</ContactPermissions>
```

This entry might be the only `ContactPermissions` entry in the file. By default, the file contains a set of `StaticHandler` entries for system permissions.

7. Stop the ContactManager server, optionally regenerate the data and security dictionaries, and then restart ContactManager, as follows:

- a. Open a command prompt in the ContactManager installation folder and then enter the following command:

```
gwb stopServer
```

- b. At a command prompt open in the ContactManager installation folder, optionally regenerate the data and security dictionaries:

```
gwb genDataDictionary
```

Regenerating the *Security Dictionary* is a good way to find out if there is a problem with the new definitions.

- c. Restart ContactManager:

```
gwb runServer
```

8. Add the new permissions to a new user role.
 - a. Log in to ContactManager as a user that has the User Admin role.
For example, user name `su` with password `gw`.
 - b. Click the **Administration** tab.
 - c. In the sidebar, click **Users & Security**→**Roles**.
 - d. In the **Roles** screen, click **Add Role**.
 - e. For **Name** enter `ABPerson Manager`.

Create new claim party and vendor permissions and associated role

About this task

In this example, you create a set of permissions that control access to contacts that have the Client tag. Because ClaimCenter assigns the Claim Party tag automatically for participants in a claim, this example also creates permissions for Claim Party tags. You then create a role that includes both sets of tags and assign it to a user.

Procedure

1. Start Guidewire Studio for ContactManager.

At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb studio
```

2. In the **Project** window, navigate to **configuration→config→extensions→Typelist** and then double-click **SystemPermissionType.ttx** to open this typelist in an editor.
3. For each of the following contact tag permission typecodes, right-click an existing typecode and choose **Add new→typecode**. Then enter the information for the new typecode.

New Code	Name	Description
claimpartytagcreate	Create contact with Claim Party tag	Permission to create a contact with a Claim Party tag
claimpartytagdelete	Delete contact with Claim Party tag	Permission to delete a contact with a Claim Party tag
claimpartytagedit	Edit contact with Claim Party tag	Permission to edit a contact with a Claim Party tag
claimpartytagview	View contact with Claim Party tag	Permission to view a contact with a Claim Party tag
clienttagcreate	Create contact with Client tag	Permission to create a contact with a Client tag
clienttagdelete	Delete contact with Client tag	Permission to delete a contact with a Client tag
clienttagedit	Edit contact with Client tag	Permission to edit a contact with a Client tag
clienttagview	View contact with Client tag	Permission to view a contact with a Client tag

4. In the **Project** window, navigate to **configuration→config→security** and double click **security-config.xml**.
5. Associate the new permissions with the Client and Claim Party tags in the **security-config.xml** file.
 - If you previously added other contact permissions, you already have a **ContactPermissions** element. In that case, add the two **ContactTagAccessProfile** elements to the existing **ContactPermissions** element.
 - If these contact permissions are the first ones you are adding, add the following new typecodes:

```
<ContactPermissions>
  <ContactTagAccessProfile tag="ClaimParty">
    <ContactCreatePermission permission="claimpartytagcreate"/>
    <ContactDeletePermission permission="claimpartytagdelete"/>
    <ContactEditPermission permission="claimpartytagedit"/>
    <ContactViewPermission permission="claimpartytagview"/>
  </ContactTagAccessProfile>
  <ContactTagAccessProfile tag="Client">
    <ContactCreatePermission permission="clienttagcreate"/>
    <ContactDeletePermission permission="clienttagdelete"/>
    <ContactEditPermission permission="clienttagedit"/>
    <ContactViewPermission permission="clienttagview"/>
  </ContactTagAccessProfile>
</ContactPermissions>
```

6. Stop the ContactManager server, regenerate the data and security dictionaries, and then restart ContactManager, as follows:
 - a. If ContactManager is running, open a command prompt in the ContactManager installation folder and then enter the following command:

```
gwb stopServer
```

- b. To ensure that your new permissions are correctly formatted, at a command prompt, navigate to the ContactManager installation folder and then regenerate the data and security dictionaries:

```
gwb genDataDictionary
```

- c. Restart ContactManager:

```
gwb runServer
```

7. Add one or more new permissions to a user role. For example:
 - a. Log in to ContactManager as a user that has the User Admin role.
For example, user name su with password gw.
 - b. Click the **Administration** tab.
 - c. In the sidebar, click **Users and Security**→**Roles**.
 - d. In the **Roles** screen, click **Add Role**.
 - e. For **Name** enter Client ContactManager.
 - f. Add the following set of permission to the role.
For each of the following permissions, click **Add** and then click in the new field. Then choose a permission from the drop-down list:
 - **Create address book contacts**
 - **Create contact with Claim Party tag**
 - **Create contact with Client tag**
 - **Delete address book contacts**
 - **Delete contact with Claim Party tag**
 - **Delete contact with Client tag**
 - **Edit address book contacts**
 - **Edit contact with Claim Party tag**
 - **Edit contact with Client tag**
 - **View address book contact search pages**
 - **View address book contacts**
 - **View contact with Claim Party tag**
 - **View contact with Client tag**
 - g. Click **Update** to add the new permissions to the role.
8. Click **Actions**→**New User**.
9. Enter the following values for the new user:

Field	Value
First name	Pat
Last name	Hu
Username	phu
Password	gw

10. Under **Roles**, click **Add**.
11. Click the empty **Name** field and choose **Client ContactManager** from the list.
12. Click **Update** to save the new user.
13. Log in as phu with password gw and ensure that this user can edit and delete contacts that have the Client tag, the Claim Party tag, and both tags.

The sample data has contacts with all these tags set. You can also log in as another user, like su, and create users with various tags for testing. To load sample data, see “Load sample data for ContactManager” on page 56.

14. Search for contacts with **Tag** specified as Vendor and verify that no contacts are returned.
15. Create a user and verify that you can assign only Claim Party and Client tags.

PolicyCenter contact security

The primary discussion of PolicyCenter security is in the *Application Guide*.

Additionally, there is an example that uses PolicyCenter contact security in “Configure ContactManager-to-PolicyCenter authentication” on page 85.

There are *ab* and *anytag* contact permissions, such as *abedit*, *abcreate*, *anytagedit*, and *anytagcreate*, that are part of a general contact security infrastructure that supports all Guidewire applications. This infrastructure also provides permission check expressions like `perm.Contact.createab` and `perm.Contact.editab`. You can configure and extend these permissions and expressions in PolicyCenter.

The following ClaimCenter topics describe how these permissions and expressions work in ContactManager and ClaimCenter:

- “ClaimCenter contact subtype and tag permissions” on page 134
- “ClaimCenter contact and tag permission check expressions” on page 135

BillingCenter contact security

BillingCenter provides contact permissions and permission check expressions to secure access to contact related tasks and screens.

BillingCenter contact-related permissions

As described at “Security” in the *BillingCenter Application Guide*, BillingCenter uses roles and permissions to limit tasks that users can perform.

For tasks related to contacts, such as editing data for an account contact, BillingCenter provides the following permissions:

Name	Code	Description
Create account contact	acctcntcreate	Permission to add a new contact to an account
Create policy contact	plcycntcreate	Permission to add a new contact to a policy period
Create producer contact	prodcntcreate	Permission to add a new contact to a producer
Delete account contact	acctcntdelete	Permission to remove a contact from an account
Delete policy contact	plcycntdelete	Permission to remove a contact from a policy period
Delete producer contact	prodcntdelete	Permission to remove a contact from a producer
Edit account contact	acctcntedit	Permission to edit information on an account contact
Edit policy contact	plcycntedit	Permission to edit information on an existing policy period contact
Edit producer contact	prodcntedit	Permission to edit information on an existing producer contact
View account contacts screen	acctcontview	Permission to view Accounts → Contacts screen
View policy contacts screen	plcycontview	Permission to view Policies → Contacts screen
View producer contacts screen	prodcontview	Permission to view Producers → Contacts screen

Additionally, there are *ab* and *anytag* contact permissions, such as *abedit*, *abcreate*, *anytagedit*, and *anytagcreate* that are part of a general contact security infrastructure that supports all Guidewire applications.

See also

- “ContactManager contact subtype and tag permissions” on page 124
- “Configure ContactManager-to-BillingCenter authentication” on page 87

BillingCenter contact and tag permission check expressions

BillingCenter uses permission check expressions to control access to contact-related screens and widgets. Each contact permission check expression is associated with a permission. For example, the `NewAccountContactPopup` widget has its `CanVisit` property set to the permission check expression `perm.AccountContact.create`. This setting allows only the users who have the `acctcntcreate` permission to see this popup.

The base configuration of BillingCenter uses the following contact-related permission check expressions:

BillingCenter Expression	Description
<code>perm.AccountContact.create</code>	Related permission is <code>acctcntcreate</code> .
<code>perm.AccountContact.delete</code>	Related permission is <code>acctcntdelete</code> .
<code>perm.AccountContact.edit</code>	Related permission is <code>acctcntedit</code> .
<code>perm.PolicyPeriodContact.create</code>	Related permission is <code>plcyntcreate</code> .
<code>perm.PolicyPeriodContact.delete</code>	Related permission is <code>plcyntdelete</code> .
<code>perm.PolicyPeriodContact.edit</code>	Related permission is <code>plcyntedit</code> .
<code>perm.ProducerContact.create</code>	Related permission is <code>prodcntcreate</code> .
<code>perm.ProducerContact.delete</code>	Related permission is <code>prodcntdelete</code> .
<code>perm.ProducerContact.edit</code>	Related permission is <code>prodcntedit</code> .

The Guidewire contact security infrastructure that provides the *ab* and *anytag* contact permissions also provides permission check expressions like `perm.Contact.createab` and `perm.Contact.editab`. You can configure and extend these permissions and expressions in BillingCenter.

Viewing permissions in the BillingCenter Security Dictionary

You can use the *Security Dictionary* to get information on the application permission keys.

For example, open the *Security Dictionary* and click the **System Permissions** filter at the top of the left pane. You see all the system permissions listed on the left by code name. If you click the permission code `acctcntcreate`, you see that it has the related application permission key **AccountContact create**, which has the Gosu check expression `perm.AccountContact.create`. You can filter the list by application permission keys, pages, system permissions, and roles.

Using the *Security Dictionary*, you can determine the following:

- The system permission related to an application permission key
- The PCF files and widgets that use an application permission key
- The roles, application permission keys, PCF pages, and widgets that use a system permission
- A list of the Gosu application permission expressions called from each PCF page
- A list of the permissions assigned to each role

See also

- “Build and view the BillingCenter Security Dictionary” on page 134
- “BillingCenter contact and tag permission check expressions” on page 133

Build and view the BillingCenter Security Dictionary

Procedure

1. At a command prompt, run the following command from the BillingCenter installation folder:

```
gwb genDataDictionary
```

This command builds the *Security Dictionary* in the following location
BillingCenter/build/dictionary/security/index.html

2. Navigate in a web browser to the location of the *Security Dictionary*. For example:
file:///C:/BillingCenter/build/dictionary/security/index.html

ClaimCenter contact security

To fully understand ClaimCenter contact security, you need also to understand ClaimCenter permissions and configuration values. See the following references:

- “Security: Roles, Permissions, and Access Controls” in the *ClaimCenter Application Guide*
- “Securing Access to ClaimCenter Objects” in the *ClaimCenter System Administration Guide*

ClaimCenter contact roles

As described in “Security: Roles, Permissions, and Access Controls” in the *ClaimCenter Application Guide*, a role is a collection of permissions.

By grouping permissions into roles, you can define a user’s authority with a few assigned roles, rather than with a much larger list of permissions. A user can have any number of roles, but has to have at least one.

You might need more granular control over who gets to view, edit, create, and delete contacts, rather than using the simple view and edit permissions. For example, you have ContactManager users that manage certain subtypes of contacts, and you want the system to enforce permissions at the contact subtype level. This role is especially important for the Service Provider Management feature, of which the list of contact subtypes—service providers—is an integral part. Only specific ContactManager users can manage the lists of these contact subtypes.

The permissions used in the base configuration of ClaimCenter for ContactManager are abview, abviewsearch, abedit, abeditpref, abcreate, abcreatepref, abdelete, abdeletepref, anytagcreate, anytagdelete, anytagedit, and anytagview.

With administrator privileges, you can assign permissions to particular users for certain contacts or contact subtypes. For example, you can grant one user the ability to manage the Auto Body Shops contact subtype and another the permission to manage other contact subtypes.

See also

- ClaimCenter contact permissions are described in more detail in “ClaimCenter contact subtype and tag permissions” on page 134

ClaimCenter contact subtype and tag permissions

ClaimCenter provides subtype and tag permissions that you can use to control access to contacts. These permissions make a distinction between local contacts and centralized, or address book, contacts. The SystemPermissionType typelist lists all the subtype and tag permissions in your system.

The following table lists the base subtype and tag permissions provided with ClaimCenter contacts:

Code	Description of Permission
anytagview	See a contact that has any contact tag.

Code	Description of Permission
anytagcreate	Create a new contact regardless of which contact tag it requires.
anytagdelete	Delete a local, unlinked contact that has any contact tag.
anytagedit	Edit a contact that has any contact tag.
ctcview	View and search local contacts.
ctccreate	Create a new, local contact.
ctccedit	Edit local contacts.
abview	View the details of contact entries retrieved from ContactManager.
abviewsearch	Search for contact entries in ContactManager.
abcreate	Create a new vendor contact in ContactManager. In the base configuration, this permission enables a ClaimCenter user to create a vendor contact and have it saved in ContactManager. Without this permission, a ClaimCenter user can create and save non-vendor contacts in ContactManager. Any vendor contacts created by a user without this permission are created in ContactManager with pending status and must be approved by a ContactManager user.
abcreatepref	Create a new preferred vendor in ContactManager.
abedit	Edit an existing vendor contact stored in ContactManager. In the base configuration, this permission enables a ClaimCenter user to edit a vendor contact and have it saved in ContactManager. Without this permission, a ClaimCenter user can edit and save non-vendor contacts in ContactManager. Any vendor contact changes by a user without this permission become pending changes in ContactManager and must be approved by a ContactManager user.
abeditpref	Edit an existing preferred vendor stored in ContactManager.

The system uses role-based security for these permissions. As described in the previous topic, to implement role-based security, a system administrator associates permissions with roles and assigns roles to users. For each role assigned, the user acquires the permissions associated with that role. For example, a role associated with the `abcreate` and `anytagcreate` permissions enables the user who has this role to create any type of contact.

The base contact and tag permissions apply across all contact subtypes. If you grant a permission to a contact type, you grant the same permissions to all that contact's subtypes.

ClaimCenter enables you to restrict permissions according to contact subtype or tag. For example, you can enable a user with `abcreate` permission to create only `PersonVendor` contacts, but not `CompanyVendor` contacts. You configure contact and tag permissions through the `SystemPermissionType` typelist and the `security-config.xml` resource.

Note: If you create a set of tag permissions for a specific tag, these permissions enable access only to contacts that have that one tag. For example, you create a set of `Vendor` tag permissions and a user has a role with only those tag permissions. That user will not be able to work with a contact that has both `Claim Party` and `Vendor` tags. You could also create a set of tag permissions for `Claim Party` tags. In that case, a user with both `Vendor` and `Claim Party` tag permissions would be able to work with contacts that have both `Vendor` and `Claim Party` tags.

See also

- For examples showing how to add typecodes to the `SystemPermissionType` typelist, set up `security-config.xml`, and implement role-based security, see “Configuring ClaimCenter contact security” on page 137.

ClaimCenter contact and tag permission check expressions

In a page configuration file (PCF), you can control permissions on specific widgets with Gosu expressions that determine if a user has permission to perform an operation.

For example, the setting for the ClaimCenter **AddressBookContactDetail** page's `canVisit` attribute is `perm.Contact.viewab(externalContact.Contact)`. This setting limits users of this page to viewing only contact

types for which they have subtype permissions and tag permissions to view the contact. Additionally, the `canEdit` attribute setting is `externalContact.Source.supportsUpdate()` and `perm.Contact.editab(externalContact.Contact)`. This setting limits a user of this page to editing only contacts that support updates and for which the user has subtype permissions and tag permissions to edit the contact.

Note: To see this file, in Guidewire Studio for ClaimCenter, press `Ctrl+Shift+N` and enter `AddressBookContactDetail`, and then click `AddressBookContactDetail.pcf` (configuration `\config\web\pcf\addressbook`) in the list of objects that the system finds.

ClaimCenter does not use tag permission checks to control access to screens or buttons. However, when a user tries to save a new contact to `ContactManager`, ClaimCenter applies the `createab` permission check expression. This permission check ensures that the user has permission to create the tag as well as the contact. If the user does not have this permission, `ContactManager` creates a pending contact that requires approval in `ContactManager` to become permanent.

Some Gosu permission check expressions require an input parameter, and some do not. The following table lists the ClaimCenter contact and tag permission check expressions:

ClaimCenter Expression	Description
<code>perm.Contact.createab</code>	Takes an input parameter that is either a Contact type or subtype or a <code>ContactTagType</code> typecode specified as an enumeration constant. Depending on the parameter, verifies that the user has the permission to create either a contact with the specified tag or a contact of the specified type.
<code>perm.Contact.createpreferredab</code>	Does not take an input parameter. Verifies that the user has permission to create the preferred contact.
<code>perm.Contact.deleteab</code>	Takes a Contact instance as an input parameter. Determines the subtype and contact tag or tags from the instance, and then verifies that the user has the contact and tag permissions to delete the contact.
<code>perm.Contact.deletepreferredab</code>	Does not take an input parameter. Verifies that the user has permission to delete the preferred contact.
<code>perm.Contact.editab</code>	Takes a Contact instance as an input parameter. Determines the subtype and contact tag or tags from the instance, and then verifies that the user has the contact and tag permissions to edit the contact.
<code>perm.Contact.editpreferredab</code>	Does not take an input parameter. Verifies that the user has permission to edit the preferred contact.
<code>perm.Contact.viewab</code>	Takes a Contact instance as an input parameter. Determines the subtype and contact tag or tags from the instance, and then verifies that the user has the contact and tag permissions to view the contact.
<code>perm.Contact.viewsearchab</code>	Does not take an input parameter. Verifies that the user has permission to edit the preferred contact in the address book.
<code>perm.Contact.createlocal</code>	Does not take an input parameter. Verifies that the user has permission to create the local contact.
<code>perm.Contact.editlocal</code>	Requires a Contact instance as an input parameter. Verifies that the user has permission to edit either an existing local contact or a user contact. To edit a user contact, the user needs edit user permission.
<code>perm.Contact.viewlocal</code>	Does not take an input parameter. Verifies that the user has permission to view and search local contact entries.

Viewing permissions in the ClaimCenter Security Dictionary

You can use the *Security Dictionary* to get information on the application permission keys. For example, if you click the **System Permissions** filter at the top of the left pane, you see all the permissions listed on the left by code name. If you click the permission code `abedit`, you see that it has the related application permission key `Contact editab`, which

has the associated Gosu check expression `perm.Contact.editab`. You can filter the list by application permission keys, pages, system permissions, and roles.

By using the *Security Dictionary*, you can determine the following:

- The system permission related to an application permission key
- The PCF files and widgets that use an application permission key
- The roles, application permission keys, PCF pages, and widgets that use a system permission
- A list of the Gosu application permission expressions called from each PCF page
- A list of the permissions assigned to each role

See also

- “Build and view the ClaimCenter Security Dictionary” on page 137

Build and view the ClaimCenter Security Dictionary

Procedure

1. At a command prompt, run the following command from the ClaimCenter installation folder:

```
gwb genDataDictionary
```

This command builds the *Security Dictionary* in the following location

`ClaimCenter/build/dictionary/security/index.html`

2. Navigate in a web browser to the location of the *Security Dictionary*. For example:

`file:///C:/ClaimCenter/build/dictionary/security/index.html`

Contact permission search configuration parameters in ClaimCenter

There are two parameters you can set in the ClaimCenter `config.xml` file to control how searching for externally stored contacts interacts with contact permission. Those parameters are `RestrictSearchesToPermittedItems` and `RestrictContactPotentialMatchToPermittedItems`. You edit `config.xml` in Guidewire Studio for ContactManager.

You can use the `RestrictSearchesToPermittedItems` configuration parameter to control the interaction between the permissions `abviewsearch` and `abview`. The `abviewsearch` permission determines which users can use the Address Book search function. However, not all users with `abviewsearch` permission also have `abview` permission. The `abview` permission enables users to view the contact’s detailed information.

If `RestrictSearchesToPermittedItems` is `false`, in response to a search, the system returns all contacts that match the search criteria. If this parameter is `true`, the system returns only contacts for which the user has view permissions. This setting also interacts with contact and tag permissions. For example, if a user can view the `Person` subtype with any tag and `RestrictSearchesToPermittedItems` is `true`, the system returns only contacts of the `Person` subtype.

Additionally, you can set the `RestrictContactPotentialMatchToPermittedItems` parameter. This parameter controls the security of potential search results. If the parameter is `true`, only the potential matches for which the user has view permissions are returned.

Configuring ClaimCenter contact security

Use the `SystemPermissionType` typelist and the `security-config.xml` configuration file to define new ClaimCenter contact security permissions. These two files enable you to create more finely grained system permissions for contact subtypes and tags.

Create contact permissions for a subtype in ClaimCenter

Procedure

1. Start Guidewire Studio for ClaimCenter.
At a command prompt, navigate to the ClaimCenter installation folder and enter the following command:

```
gwb studio
```

2. In the **Project** window, navigate to **configuration→config→Extensions→Typelist**.
3. Double-click **SystemPermissionType.ttx** to open this typelist in the editor.
4. For each of the following contact permission typecodes, right-click an existing typecode and choose **Add new→typecode**. Then enter the information for the new typecode.

New Code	Name	Description
companyvendorcreate	Create company vendor contacts	Permission to create company vendor contacts
companyvendoredit	Edit company vendor contacts	Permission to edit company vendor contacts
companyvendordelate	Delete company vendor contacts	Permission to delete company vendor contacts
companyvendorview	View company vendor contacts	Permission to view company vendor contacts
personvendorcreate	Create person vendor contacts	Permission to create person vendor contacts
personvendoredit	Edit person vendor contacts	Permission to edit person vendor contacts
personvendordelate	Delete person vendor contacts	Permission to delete person vendor contacts
personvendorview	View person vendor contacts	Permission to view person vendor contacts

5. In the **Project** window, navigate to **configuration→config→security** and double-click **security-config.xml**.
6. Associate the new permissions with a Contact subtype in the **security-config.xml** file.
For example, add the new typecodes for the **Vendor** and **VendorPerson** contact subtype permissions to the **ContactPermissions** element as follows:

```
<ContactPermissions>
...
  <ContactSubtypeAccessProfile entity="CompanyVendor">
    <ContactCreatePermission permission="companyvendorcreate"/>
    <ContactEditPermission permission="companyvendoredit"/>
    <ContactDeletePermission permission="companyvendordelate"/>
    <ContactViewPermission permission="companyvendorview"/>
  </ContactSubtypeAccessProfile>
  <ContactSubtypeAccessProfile entity="PersonVendor">
    <ContactCreatePermission permission="personvendorcreate"/>
    <ContactEditPermission permission="personvendoredit"/>
    <ContactDeletePermission permission="personvendordelate"/>
    <ContactViewPermission permission="personvendorview"/>
  </ContactSubtypeAccessProfile>
...
</ContactPermissions>
```

7. Stop the ClaimCenter server, regenerate the data and security dictionaries, and then restart ClaimCenter, as follows:
 - a. If ClaimCenter is running, open a command prompt in the ClaimCenter installation folder and then enter the following command:

```
gwb stopServer
```

- b. To verify that the new permissions are correct, at a command prompt, navigate to the ClaimCenter installation folder and regenerate the data and security dictionaries:

```
gwb genDataDictionary
```

- c. After you get a successful build of the dictionaries, restart ClaimCenter:

```
gwb runServer
```

8. Add the new permissions to a user role. For example:

- a. Log in to ClaimCenter as a user that has the User Admin role, such as user name su with password gw.
- b. Click the **Administration** tab.
- c. Click **Users & Security**→**Roles** in the sidebar.
- d. Click **Add Role**.
The New Role screen opens.
- e. For **Name**, enter Vendor Admin.
- f. For **Description**, enter Manages vendor contacts. Requires Clerical role as well.
- g. Click **Add** to add a new permission.
- h. Click the new field and choose the **Create company vendor contacts** permission from the drop-down list.
- i. Repeat Step g and Step h for each of the following permissions, substituting the actual permission for **Create company vendor contacts** in Step h:
 - **Create person vendor contacts**
 - **Delete company vendor contacts**
 - **Delete person vendor contacts**
 - **Edit company vendor contacts**
 - **Edit person vendor contacts**
 - **View company vendor contacts**
 - **View person vendor contacts**
- j. Click **Update** to add the new role.


9. Remove the abview permission from the Clerical role.

Note: Removing this permission is a quick way to test the new Vendor Admin role. With abview removed, using the Clerical role in conjunction with Vendor Admin makes it possible to limit users with the two roles from viewing non-vendor contacts. However, after this change, users who have the Clerical role and no other will be unable to view any contacts. For this change to be permanent, another role with abview permission would need to be created and then used in conjunction with the Clerical role for all previous Clerical users.

- a. Click the **Administration** tab.
- b. Click **Users & Security**→**Roles** in the sidebar.
- c. Click **Clerical** and then click **Edit**.
- d. Click the **Code** column heading to list abview at the top.
- e. Click the check box for abview, and then click **Remove**.
- f. Click **Update** to save your changes.

10. Create a new user for the role.

- a. Click **Actions**→**New User**.
The **New User** screen opens.
- b. For **First name**, enter Devra.
- c. For **Last name**, enter Rajan.
- d. For **User name**, enter drajan.
- e. For **Password** and **Confirm Password**, enter gw.
- f. Under **Roles**, click **Add**, and then click the **Name** field. Choose **Vendor Admin** from the drop-down list.

- g. Under **Roles**, click **Add** again, and then click the **Name** field. Choose **Clerical** from the drop-down list.
 - h. Under **Groups**, click **Add** and then click the **Group** field. Choose a group from the list, such as the sample data group **Western Regional Claims Center**.
 - i. Click **Update** to create the new user.
11. Log out.
For example, if you logged in as su, on the Options menu , click **Log Out Super User**.
12. Log in as drajan with password gw.
13. Verify that this user can create a new CompanyVendor or PersonVendor subtype and have it saved in ContactManager without being pending.
14. Verify that this user can edit and delete contacts of these types and have the changes saved in ContactManager without the contacts being made pending.
15. Verify that in an Address Book search, this user can click in the search results and see details only for CompanyVendor and PersonVendor contact subtypes and not for non-vendor subtypes. For example, clicking a contact of type Company or Person displays a message saying that you do not have permission to view the contact detail popup.

Create claim party and vendor tag permissions in ClaimCenter

About this task

The base application comes with a set of tag permissions that permit a user to create, delete, edit, and view all tags. You can add permissions that control access to tags at a more granular level. This topic shows how to create a set of tag permissions in ClaimCenter.

Note: In this example, you create a set of Vendor tag permissions. These permissions enable a user to see and work with contacts that have only the Vendor tag. If the contact has any other tags, these permission do not enable working with that contact. For example, a user who has a single role with only Vendor tag permissions is not able to work with a contact that has both Claim Party and Vendor tags. You could create a set of tag permissions for Claim Party tags and add them to the role that has the Vendor tag permissions. A user with that role would be able to work with contacts that have both Vendor and Claim Party tags. However, that user would not be able to work with contacts that have both Vendor and Client tags. For more information on the application tags in the base configuration, see “ClaimCenter contact subtype and tag permissions” on page 134.

Procedure

1. Start Guidewire Studio for ClaimCenter.

At a command prompt, navigate to the ClaimCenter installation folder and enter the following command:

```
gwb studio
```

2. In the **Project** window, navigate to **configuration**→**config**→**Extensions**→**Typelist**.
3. Double-click **SystemPermissionType.etx** to open this typelist in an editor.
4. For each of the following contact permission typecodes, right-click an existing typecode and choose **Add new**→**typecode**. Then enter the information for the new typecode.

New Code	Name	Description
claimpartytagcreate	Create contact with Claim Party tag	Permission to create a contact with a Claim Party tag
claimpartytagdelete	Delete contact with Claim Party tag	Permission to delete a contact with a Claim Party tag
claimpartytagedit	Edit contact with Claim Party tag	Permission to edit a contact with a Claim Party tag
claimpartytagview	View contact with Claim Party tag	Permission to view a contact with a Claim Party tag

New Code	Name	Description
vendortagcreate	Create contact with Vendor tag	Permission to create a contact with a Vendor tag
vendortagdelete	Delete contact with Vendor tag	Permission to delete a contact with a Vendor tag
vendortagedit	Edit contact with Vendor tag	Permission to edit a contact with a Vendor tag
vendortagview	View contact with Vendor tag	Permission to view a contact with a Vendor tag

5. In the **Project** window, navigate to **configuration→config→security** and double-click **security-config.xml**.

6. Associate the new permissions with a contact tag in the **security-config.xml** file.

For example, add the new typecodes for the Vendor tag permissions to the **ContactPermissions** element as follows:

```
<ContactPermissions>
...
  <ContactTagAccessProfile tag="Vendor">
    <ContactCreatePermission permission="vendortagcreate"/>
    <ContactDeletePermission permission="vendortagdelete"/>
    <ContactEditPermission permission="vendortagedit"/>
    <ContactViewPermission permission="vendortagview"/>
  </ContactTagAccessProfile>
...
</ContactPermissions>
```

7. Stop the ClaimCenter server, regenerate the data and security dictionaries, and then restart ClaimCenter, as follows:

- a. If ClaimCenter is running, open a command prompt in the ClaimCenter installation folder and then enter the following command:

```
gwb stopServer
```

- b. To verify that the new permission are correctly formatted, at a command prompt open in the ClaimCenter installation folder, regenerate the data and security dictionaries:

```
gwb genDataDictionary
```

- c. Restart ClaimCenter:


```
gwb runServer
```

8. Add the new permissions to a user role. For example:

- a. Log in to ClaimCenter as a user that has the User Admin role, such as user name **su** with password **gw**.
- b. Click the **Administration** tab.
- c. Click **Users & Security→Roles** in the Sidebar.
- d. Click **Add Role**.
The New Role screen opens.
- e. For **Name**, enter **Vendor & Claim Party Tag Admin**.
- f. For **Description**, enter **Manages contacts with Vendor and Claim Party tags. Requires Clerical role as well**.
- g. Click **Add** to add a new permission.
- h. Click the new field and choose the **Create contact with Vendor tag** permission from the drop-down list.

- i. Repeat Step g and Step h for each of the following permissions, substituting the actual permission for **Create contact with Vendor tag** in Step h:
 - **Create address book contact**
 - **Create contact with Claim Party tag**
 - **Delete address book contact**
 - **Delete contact with Claim Party tag**
 - **Delete contact with Vendor tag**
 - **Edit address book contact**
 - **Edit contact with Claim Party tag**
 - **Edit contact with Vendor tag**
 - **View address book contact**
 - **View contact with Claim Party tag**
 - **View contact with Vendor tag**
 - j. Click **Update** to add the new role.
9. Remove the anytagview permission from the Clerical role.

Note: Removing this permission is a quick way to test the new Vendor & Claim Party Tag Admin role. However, after this change, users who have only the Clerical role and no other will be unable to view contacts that have tags. For contacts stored in ContactManager, all contacts must have tags, so they would not be visible. For this change to be permanent, another role with anytagview permission would need to be created and then used in conjunction with the Clerical role for all previous Clerical users.

 - a. Click the **Administration** tab.
 - b. Click **Users & Security**→**Roles** in the Sidebar.
 - c. Click **Clerical** and then click **Edit**.
 - d. Click the **Code** column heading to list anytagview on the first page.
 - e. Click the check box for anytagview, and then click **Remove**.
 - f. Click **Update** to save your changes.
 10. Create a new user for the role.
 - a. Click **Actions**→**New User**.
The **New User** screen opens.
 - b. For **First name**, enter Mira.
 - c. For **Last name**, enter No1o.
 - d. For **User name**, enter mno1o.
 - e. For **Password** and **Confirm Password**, enter gw.
 - f. Under **Roles**, click **Add**, and then click the **Name** field. Choose **Vendor & Claim Party Tag Admin** from the drop-down list.
 - g. Under **Roles**, click **Add** again, and then click the **Name** field. Choose **Clerical** from the drop-down list.
 - h. Under **Groups**, click **Add** and then click the **Group** field. Choose a group from the list, such as the sample data group **Eastern Regional Claims Center**.
 - i. Click **Update** to create the new user.
 11. Log out.
For example, if you logged in as su, on the Options menu , click **Log Out Super User**.
 12. Log in as mno1o with password gw.
 13. Verify that this user can find contacts with Vendor and Claim Party tags set, or with only one tag or the other set. If necessary, either create those contacts in ContactManager or import ContactManager sample data.
 14. Verify that this user can create, edit, and delete contacts with these tags and have the changes saved in ContactManager without the contacts being made pending.

Extending the contact data model

In the base configuration, ContactManager provides a set of `ABContact` entities. The core applications provide a corresponding set of `Contact` entities. You can extend these entities by adding subtypes, and you can make changes to the user interface to support the new subtypes.

Overview of contact entities

Before extending the contact data model, you need to know what is provided in the ContactManager and core application base configurations. Additionally, there are guidelines you must follow when planning to extend these data models.

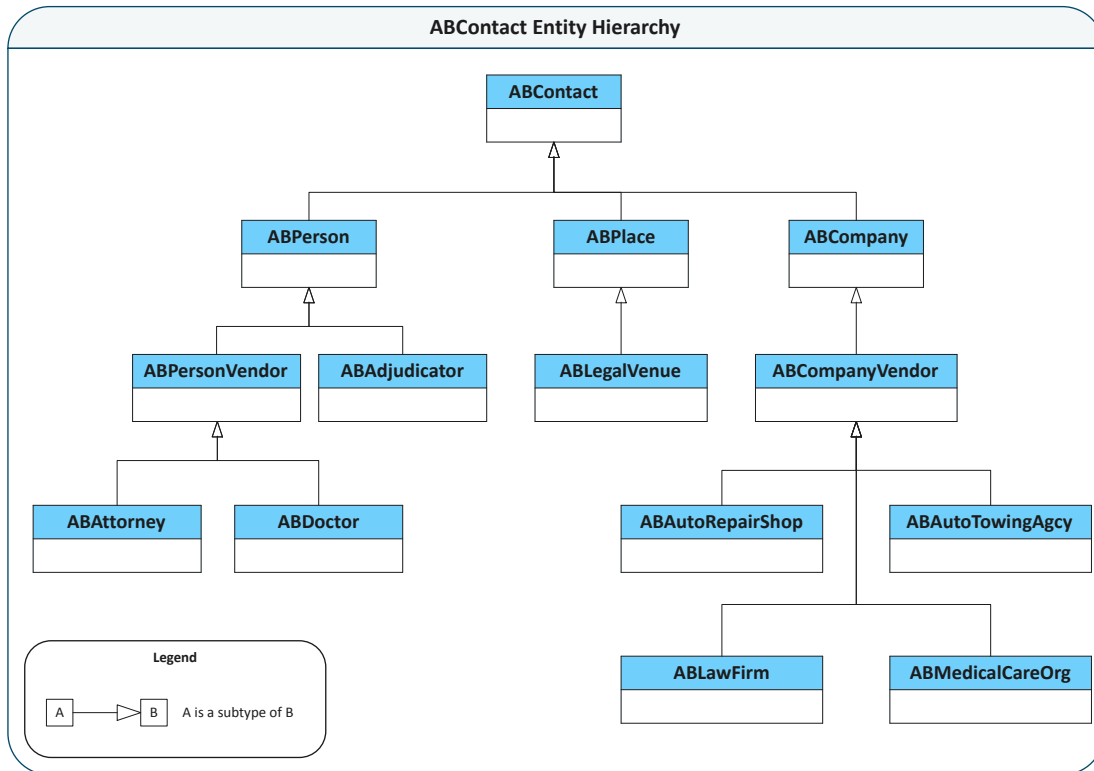
ABContact data model

The `ABContact` entity is the ContactManager data model entity used in both client and vendor contact management. `ABContact` has an entity hierarchy and a set of relationships to other entities.

ABContact entity hierarchy

The `ABContact` entity has three primary subtypes for various types of contacts, `ABPerson`, `ABCompany`, and `ABPlace`. These subtypes have additional subtypes, like `ABAdjudicator`, `ABCompanyVendor`, `ABLegalVenue`, and so on. The following figure shows the `ABContact` entity hierarchy. This entity hierarchy has a parallel in the `Contact` entity hierarchy in the core applications. See “ContactManager and core application contact entity hierarchies” on page 147.

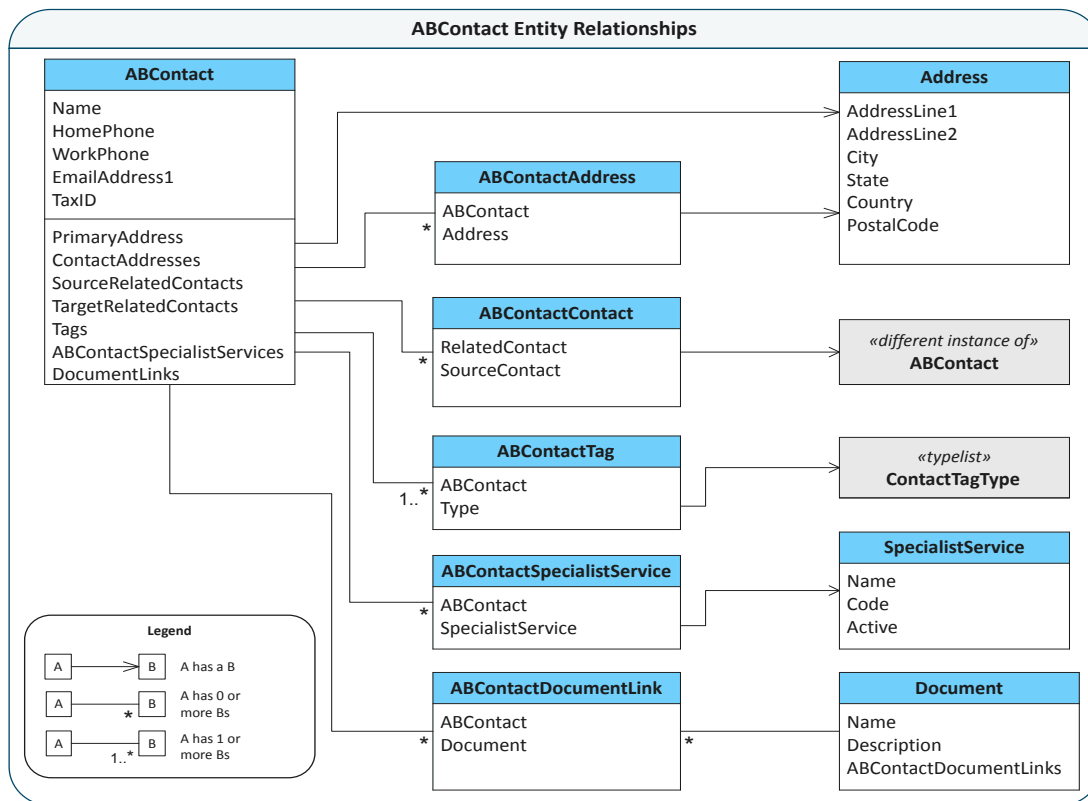
Note: This figure omits the subtypes `ABUserContact`, `ABPolicyCompany`, and `ABPolicyPerson`. Although they exist in the base configuration, these entities are not intended to be used in `ContactManager`. If you need to store address and phone information for a `ContactManager` user, use the `UserContact` entity as described in “Contact data model” on page 145.



ABContact entity relationships

The `ABContact` entity has fields for name, phone number, email address, and so forth. For many vendor contacts it is necessary to maintain a tax ID for tax reporting. The `ABContact` entity tracks this data as well. Some of this data is stored in other entities.

In the base configuration, a contact must have at least one tag. A contact can have multiple addresses, related contacts, tags, and services. References to those entities are handled with arrays of join entities, such as `ContactAddress` for addresses and `ContactContact` for related contacts, and derived properties, as shown in the following figure:



The physical location, the address, of a contact is not maintained in the ABContact entity. Instead, the ABContact entity references another entity, the Address entity, which maintains the contact's street or mailing address. An ABContact entity can reference a primary address and, through the ABContactAddress entity, other secondary addresses.

Contacts can have relationships with other contacts. For example, an ABPerson contact might be employed by a particular ABCompany contact. The ABContactContact entity maintains data about the relationships a contact can have with other contacts.

Note: For simplicity, the diagram shows ABContactContact connecting to a different instance of ABContact. However, ABContactContact can also point back to the original contact. For example, you can be your own primary contact.

Contacts can have tags, like Client and Vendor, and specialist services that the contact provides, like carpentry or independent appraisal. An ABContact entity references its tags by using an array of ABContactTag entities. An ABContact entity references its services by using an array of ABContactSpecialistService entities.

Contact data model

A Contact is the Guidewire core application data model entity used in both client and vendor contact management. In the base configuration, this entity is the core application equivalent of the ContactManager entity ABContact, described previously in "ABContact data model" on page 143.

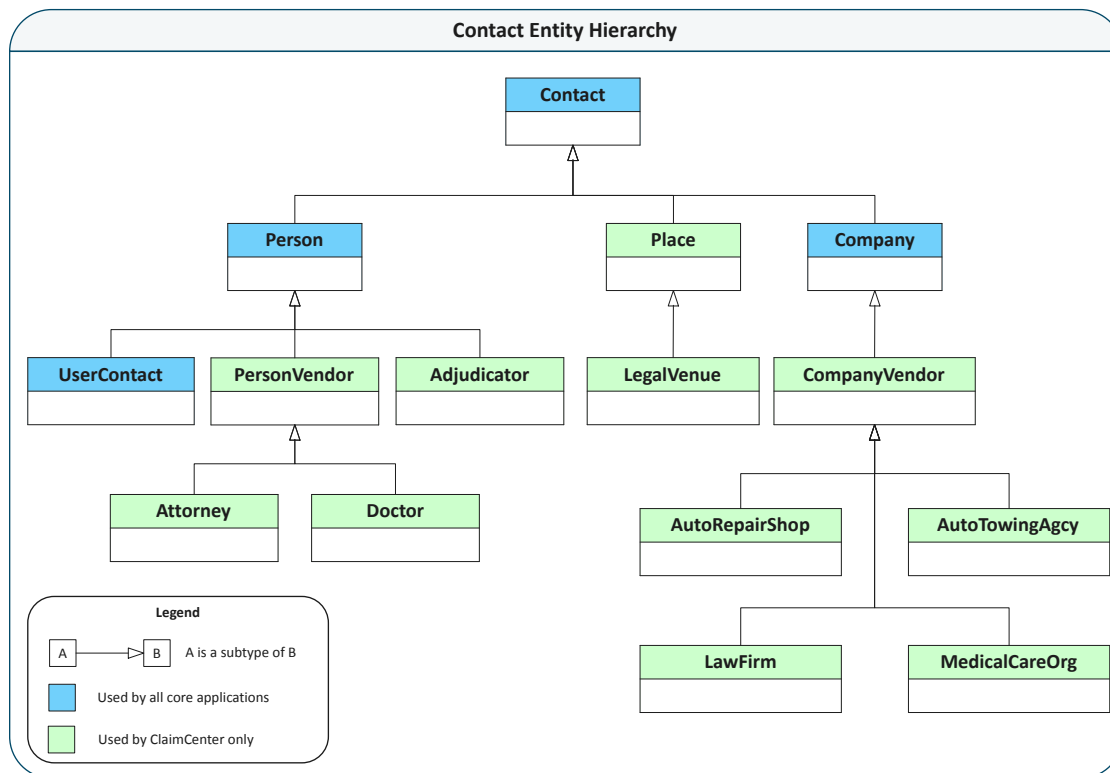
The Contact entity has an entity hierarchy and a set of relationships to other entities.

Contact entity hierarchy

The Contact entity has subtypes for various types of contacts, like Person, Company, and Place. In their base configurations, PolicyCenter and BillingCenter use just the Person and Company subtypes to link with ContactManager.

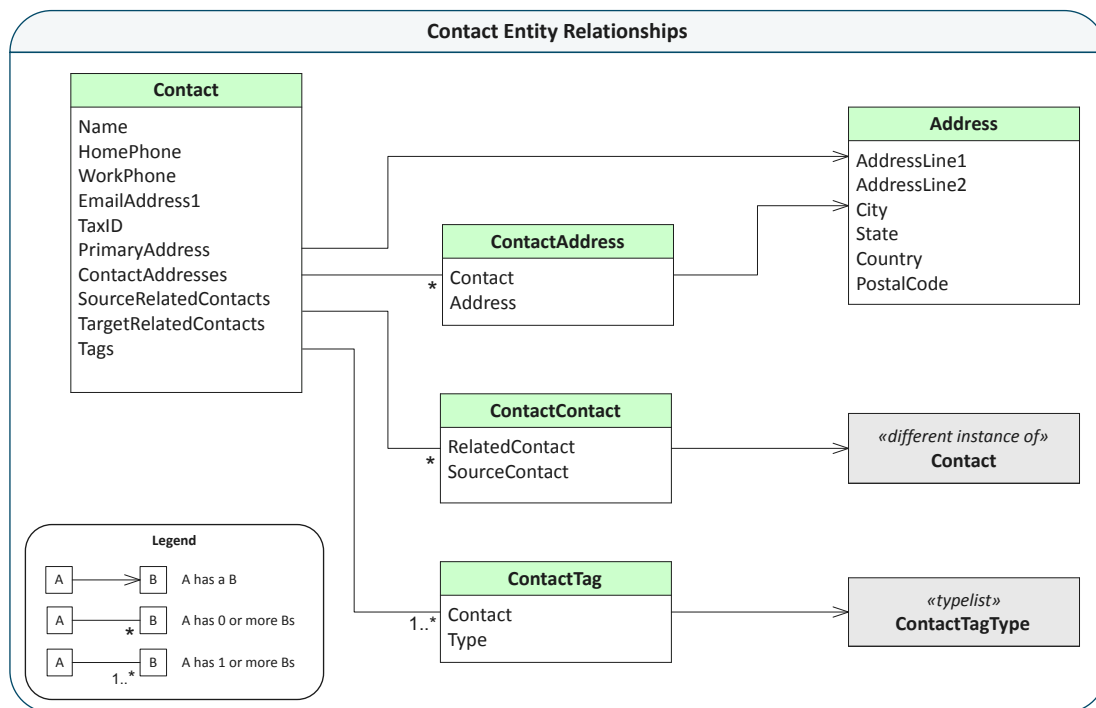
In ClaimCenter, these subtypes have additional subtypes, like *Adjudicator*, *CompanyVendor*, *LegalVenue*, and so on. The following figure shows the *Contact* entity hierarchy. This entity hierarchy has a parallel in the *ABContact* entity hierarchy in *ContactManager*. See “*ContactManager* and core application contact entity hierarchies” on page 147.

Note: The following diagram includes a special *Contact* subtype, *UserContact*. This entity is used by the *User* entity, which represents a user of the application. The *User* entity has a foreign key to *UserContact* to store data like the user’s address and phone number, which the application can display in its user interface. Additionally, the *UserContact* entity makes it possible to do things like proximity search to assign users to claims. *UserContact* entities are not intended to be used as either vendor contacts or client contacts, and they do not link to *ContactManager*.



Contact entity relationships

The *Contact* entity is associated with other entities. A contact can have multiple addresses, related contacts, and tags. A *Contact* entity must have at least one tag. Except for the primary address, references to those entities are handled with arrays of join entities. For example, there is *ContactAddress* for addresses and *ContactContact* for related contacts, as shown in the following figure:



The physical location, the address, of a **Contact** is stored in the **Address** entity, which maintains the contact's street or mailing address. A **Contact** can reference a primary address and, through the **ContactAddress** entity, other secondary addresses.

Contacts can have relationships with other contacts. For example, a **Person** might be employed by a particular **Company**. The **ContactContact** entity maintains data about the relationships a contact can have with other contacts.

Note: For simplicity, the diagram shows **ContactContact** connecting to a different instance of **Contact**. However, **ContactContact** can also point back to the original contact. For example, you can be your own primary contact.

Contacts can have tags, like **Client** and **Vendor**, and specialist services that the contact provides, like carpentry or independent appraisal. A **Contact** entity references its tags by using the **ContactTag** entity. The relationship between a contact and its services are maintained by **ContactManager**, which is why there is no property accessing services in the **Contact** entity relationship model.

See also

- "Contact tags" on page 191

ContactManager and core application contact entity hierarchies

The Guidewire core application data models and the **ContactManager** data model both group and classify contact data as hierarchies. By default, **ContactManager** provides a contact entity hierarchy that supports both the **Client Data Management** add-on module and **ClaimCenter** vendor data management. Additionally, **ContactManager** entities have the prefix **AB** (for *Address Book*).

PolicyCenter and **BillingCenter** use only part of the hierarchy for client data management: **Contact**, **Person**, **Company**, and **UserContact**.

ClaimCenter uses the entire hierarchy. In the default application, all the **ClaimCenter** contact types but **UserContact** appear in **ContactManager** with **AB** prefixes. The following table shows the **ClaimCenter** hierarchy and the supporting hierarchy in **ContactManager** supplied in the base products:

ClaimCenter	ContactManager
Contact	ABContact
Person	ABPerson
Adjudicator	ABAdjudicator
PersonVendor	ABPersonVendor
Attorney	ABAttorney
Doctor	ABDoctor
UserContact	
Company	ABCompany
CompanyVendor	ABCompanyVendor
AutoRepairShop	ABAutoRepairShop
AutoTowingAgcy	ABAutoTowingAgcy
LawFirm	ABLawFirm
MedicalCareOrg	ABMedicalCareOrg
Place	ABPlace
LegalVenue	ABLegalVenue

Each core application Contact entity and subentity has a corresponding ABContact entity or subentity in ContactManager, with the exception of UserContact.

Note: While there is an ABUserContact entity in the ABContact hierarchy, this entity is not used, even for ContactManager users. UserContact is used both in core applications and ContactManager solely to provide address and other contact information for a User. A core application UserContact does not link to ContactManager. UserContact is never used for vendor or client information.

When ContactManager creates a contact, an instance of an ABContact entity or subentity, ContactManager also creates a unique identifier for that instance. This unique identifier is used both by core applications and by ContactManager to ensure that each application is referencing the same contact. In the core applications, AddressBookUID is the field used to uniquely identify a contact that is stored in ContactManager. In ContactManager, the corresponding field is LinkID.

For a contact stored both in ContactManager and in a core application, ContactManager sends the unique identifier in the LinkID field to each core application that uses the contact. The core application stores this unique identifier in the AddressBookUID field of the contact. If a contact in a core application has a non-null value in its AddressBookUID field, that value means that the contact is stored in ContactManager. Otherwise, the AddressBookUID field is null, meaning that the contact is stored only locally in the core application, and not in ContactManager.

To enable all applications to reference the same contact, you must not change an AddressBookUID or a LinkID field. Guidewire designed the data models to enable the Guidewire core applications and ContactManager to be compatible. If you make a contact-related extension to a Guidewire core application's data model, you typically extend the ContactManager data model as well to reflect the change. You must extend both applications if you want contact information that is captured, stored, and used in one application to be available to the other. If you have more than one core application installed, you might have to make the same extension in your other core applications as well.

Mapping of entities between applications ensures that contact records are stored with the correct entity in both the Guidewire core application and ContactManager. Mapping is not the same as linking contacts or *synchronizing* them, which determines if the records are the same between the two applications. Linking and synchronizing are separate operations from mapping.

See also

- “Linking and synchronizing contacts” on page 195
- “ContactManager link IDs and comparison to other IDs” on page 260

General guidelines for extending contacts

With some restrictions, you can customize the contact entity hierarchy by adding subtypes or custom fields. You cannot make any of the following hierarchy modifications:

- You cannot delete or move the root entity **Contact**.
- You cannot delete or move the three subtypes **Person**, **Company**, and **Place**.
- You cannot create a contact entity that is a peer to **Contact**.
- You cannot create a contact entity that is a parent to **Person**, **Company**, or **Place**.

You can add a field to any of these entities. If you add a field to an entity that is higher in the hierarchy, all entities below it inherit the field. Before adding a field, ensure that the field makes sense for all the entity’s subtypes.

You can create a peer to **Person**, **Company**, or **Place**, and you can modify these three entities. A peer entity to one of these three entities is a subtype of **Contact**. You can also create a new subtype and modify the other subtypes.

If you have integrated your Guidewire core application with **ContactManager**, you typically extend both the core application and the **ContactManager** hierarchies to mirror each other. Most contacts require central management. If you have a contact subtype that does not require central management, you can create that type just in the Guidewire core application and not in **ContactManager**.

The Guidewire core applications provide the Gosu class **ContactMapper** that you use when extending the **Contact** data model for coordination with **ContactManager**. The corresponding class you use in **ContactManager** is **gw.webservice.ab.ab1000.abcontactapi.ContactMapper**. Each Guidewire core application also has a pair of XML configuration files for mapping contact names and typelists to and from **ContactManager**.

The matching and searching functions for contacts require each subtype to have a collection of fields that makes it unique.

See also

- “Working with contact mapping files” on page 150
- “Linking and synchronizing contacts” on page 195

Deciding whether to create a subtype

Plan carefully before manipulating your contact hierarchy. A new subtype inherits the attributes and matching behavior of its supertype. Create a new subtype only if you need to treat one set of contacts differently from another. As much as possible, limit the number of subtypes you need to represent your contacts.

If you have **ClaimCenter** installed, consider using vendor services rather than creating new vendor contact subtypes.

Another alternative is to create your own tags and apply them to contacts.

The main reason to limit creation of new subtypes is that the more you specialize the subtype hierarchy, the more restrictive and the less flexible your model becomes. For example, there are subtypes for **AutoRepairShop** and **AutoTowingAgcy**. If you work with an auto repair shop that also does towing, you cannot create a single contact that does both. However, you can add a service for Towing to an auto repair shop contact.

If a contact has different roles or skill sets, you can use one subtype and add contact tags or a specialty array to represent those skills and specialties.

Additionally, each time you create a new subtype, you must modify PCF files to support both creating the subtype and searching for it. You might also need to make supporting modifications to the screens that reference contacts.

After making any data model modification, you must refresh the application and possibly the web services as well.

See also

- “Contact tags” on page 191
- “Refresh applications after contact data model changes” on page 150
- For examples of modifications to contact entities and the class hierarchy, see the following topics:
 - “Adding a field to a contact subtype” on page 152
 - “Adding a vendor contact subtype” on page 158
 - “Extending contacts with an array” on page 171

Refresh applications after contact data model changes

About this task

After making a data model modification, you must refresh the application in which you made them. If you change the data model in ContactManager, you must also refresh core application web services.

Procedure

1. Stop the applications in which you have made the data model modifications.
2. Optionally regenerate the *Data Dictionary* for the application by running `gwb genDataDictionary` from a command prompt.
This step enables you to verify that your changes work before rebuilding the application.
3. If you have made a data model change to ContactManager, start ContactManager to refresh its web services.
4. Start Guidewire Studio for the core application and refresh the web service collection for that web service.
5. Start the applications and confirm your changes.

Limitations on configuring contact entity extensions

There are limitations on the kinds of things you can do with contact type extensions. Your subtype can inherit from only a single parent entity—multiple inheritance is not supported. For example, you could represent an adjudication practice with a single owner-proprietor either as a *Company* or as an *Adjudicator*. You cannot create a subtype that inherits the fields of both entities. To create the subtype you want, you can select one entity or the other and subtype it as something like *Practice*. Then, you add the fields that are missing from the other entity because of single inheritance.

Unless you restrict the visibility of some subtypes through configuration, they can still appear in search results because searches return all subtypes. This behavior has special implications if your installation is integrated with ContactManager. For example, suppose that you configure ClaimCenter to make *Adjudicator* not a choice in the user interface. However, in the ContactManager database there are a number of *ABAdjudicator* contact entities. Searching the **Address Book** for a *Person* returns *ABAdjudicator* matches if they exist in ContactManager.

See also

- For information about restricting access to contact subtypes, see “Securing access to contact information” on page 121.

Working with contact mapping files

If you extend your Guidewire core application’s contact data model, for ContactManager to be able to work with the extension, you must also make a matching extension in ContactManager. You then map the entities to each other in both the Guidewire core application and in ContactManager.

A Guidewire core application uses the *ContactMapper* class to map the fields of contact entities sent to and received from ContactManager. There is a mapping class in each Guidewire core application and in ContactManager. The mapping classes, set up by default for the base application *Contact* types and the base ContactManager *ABContact* types, match Guidewire core application entity types to entity types in ContactManager.

IMPORTANT When PolicyCenter receives an Address update from ContactManager, PolicyCenter uses the class `gw.webservice.pc.pc1000.contact.AddressDataCopier` to populate the address for the contact. PolicyCenter also uses `AddressDataCopier` to update linked addresses. If you make extensions to the Address object, ensure that you make the same updates in both `ContactMapper` and `AddressDataCopier`.

If you extend the contact data model, you must edit these classes and map every field for the new entity that you want to send and receive. In each `ContactMapper` class in the Guidewire core application and in `ContactManager`, you map both incoming and outgoing entities. If you leave a field out, it is not processed.

Additionally, you might need to map `Contact` subtypes and typecodes whose names in a Guidewire core application are different from the names in `ContactManager`. For example, Guidewire core applications use the `Person` subtype, which in `ContactManager` is `ABPerson`. All contact data passed through the `ContactManager` web services use the `ContactManager` domain namespace. Therefore, it is up to the core applications to map names, like `Contact`, between the core application domain namespace and the `ContactManager` domain namespace, which uses `ABContact`. There is a Gosu class in each Guidewire core application for this purpose as described in “Core application mapping” on page 151.

Note: There might be situations in which you add a contact subtype to a Guidewire core application and not to `ContactManager`. If you have a contact subtype that does not require central management, you can create that type in the Guidewire core application only and not use the mapping files.

ContactManager mapping

In `ContactManager`, you use the class `gw.contactmapper.ab1000.ContactMapper` to map contact entity fields between `ContactManager` and `ClaimCenter` or one of the other Guidewire core applications.

You can access this class in Guidewire Studio for `ContactManager` from the **Project** window. Navigate to **configuration**→**gsrsrc**, and then open `gw.contactmapper.ab1000.ContactMapper`.

For reference information on this class, see “`ContactManager ContactMapper` class” on page 276.

Core application mapping

In core applications, you use the class `gw.contactmapper.ab1000.ContactMapper` to map the fields of contact entities between `ClaimCenter` and `ContactManager`.

You can access this class in Guidewire Studio from the **Project** window. Navigate to **configuration**→**gsrsrc** and open `gw.contactmapper.ab1000.ContactMapper`.

For reference information on this class, see “`ContactMapper` class” on page 275.

While `ContactMapper` maps the fields of the entities, it does not map differing entity names. `ContactManager` contact entity names, such as `ABContact` or `ABPerson`, typically start with `AB`. Core application contact entity names, such as `Contact` or `Person`, typically do not start with `AB`. Additionally, there can be differences in typecodes between the entities. Core applications do their own name mapping—no changes are needed in `ContactManager`, and there is no equivalent class in `ContactManager`.

To support mapping of differing entity names and typecodes, the core applications use the following name mapping classes:

ClaimCenter

`gw.contactmapper.ab1000.CCNameMapper`

PolicyCenter

`gw.contactmapper.ab1000.PCNameMapper`

BillingCenter

`gw.contactmapper.ab1000.BCNameMapper`

See also

- “Map the new subtype names in `ClaimCenter`” on page 160

Extending the client data model

The techniques required to extend the client data model are essentially the same as those required to extend the vendor data model.

The primary differences are:

- The client data model includes only the **Contact**, **Person**, and **Company** entities.
- All three core Guidewire applications use client data. If you have more than one core application installed, you must apply your extension changes to all the applications. For example, you have installed the entire InsuranceSuite set of applications. If you extend the client data model in PolicyCenter and ContactManager, you must also extend it in ClaimCenter and BillingCenter.
- PolicyCenter and BillingCenter PCF files are different from those of ClaimCenter.

See also

- “Extending the vendor contact data model” on page 152
- “Overview of contact entities” on page 143

Extending the vendor contact data model

You can extend the vendor contact data model used in ClaimCenter to add your own subtypes or fields. The extension examples show some common configurations you can use to extend your own installation.

Adding a field to a contact subtype

You can extend a contact with a new field.

This example is a multi-step process divided into a series of topics. In this example, you add a new **BoardCertified_Ext** field to the **Doctor** subtype in ClaimCenter and to the **ABDoctor** subtype in ContactManager. You then update the necessary files and screens to make the new field usable across the applications.

Add a field to the Doctor subtype in ClaimCenter

As part of an example of adding a field to a contact subtype, extend the **Doctor** entity in ClaimCenter.

Before you begin

If you have not done so already, follow the instructions for installing ContactManager in “Installing ContactManager” on page 55. Additionally:

- You must have integrated ContactManager with ClaimCenter as described in “Integrating ContactManager with Guidewire core applications” on page 59.
- If you do not have any users or claims defined, you can load sample data for both applications. See “Load sample data for ContactManager” on page 56.

About this task

This step is part of the example “Adding a field to a contact subtype” on page 152.

Procedure

1. Start Guidewire Studio™ for ClaimCenter.

At a command prompt, navigate to the ClaimCenter installation folder and enter:

```
gwb studio
```

2. In the **Project** window, navigate to **configuration**→**config**→**Extensions**→**Entity**.
3. Double-click **Doctor.eti** to open it in the Entity editor.

4. Click **subtype** at the top of the **Element** hierarchy.
5. Click the drop-down list for **+** and choose **column**.
6. Enter the following values for the new column:

Name	Value
name	BoardCertified_Ext
type	bit
nullok	false
desc	Is the doctor Board certified in the specialty?
default	false

7. Regenerate the *Data Dictionary* to ensure that your changes were correct.
At a command prompt, navigate to the ClaimCenter installation folder and enter:

```
gwb genDataDictionary
```

Next steps

“Add the new field to the ContactMapper class in ClaimCenter” on page 153

Add the new field to the ContactMapper class in ClaimCenter

For ClaimCenter to be able to send a new **Contact** field to ContactManager and receive updates for the field, you must add it to the ContactMapper class.

Before you begin

Complete “Add a field to the Doctor subtype in ClaimCenter ” on page 152.

About this task

For more information on the ContactMapper class, see “ContactMapper class” on page 275.

Procedure

1. In Guidewire Studio™ for ClaimCenter, press **Ctrl+N** and enter **ContactMapper**.
2. Double-click the **ab1000** version of the class in the search results to open it in the Gosu editor.
3. Press **Ctrl+F** and search for the following line of code:

```
fieldMapping(Doctor#MedicalLicense),
```

4. On a new line after the **MedicalLicense** line, enter **fieldMapping(Doctor#Board** and press **Ctrl+Space** to complete the property name, and then add a comma. The new line will be:

```
fieldMapping(Doctor#BoardCertified_Ext),
```

5. If necessary, stop ClaimCenter as follows:
 - a. Open a command prompt in the ClaimCenter installation folder and then enter the following command:

```
gwb stopServer
```

- b. Wait to restart the ClaimCenter application and pick up the data model changes until you have made the ClaimCenter user interface changes in a later topic.

Next steps

“Add the BoardCertified field to the Address Book user interface” on page 154

Add the BoardCertified field to the Address Book user interface

In this step, you modify the ClaimCenter Address Book screen so you can see the medical specialty field with the Doctor subtype.

Before you begin

Complete “Add the new field to the ContactMapper class in ClaimCenter” on page 153.

About this task

Update the input sets used by the detail view.

Procedure

1. Navigate in the **Project** window to **configuration→config→Localizations→Resource Bundle 'display'** and double-click **display.properties** to open this file in the editor.
2. Press **Ctrl+F** and search for the following entry in the file:

```
Web.ContactDetail.Doctor.MedicalLicense
```

3. Add a line under that entry, and then enter the following key and value:
`Web.ContactDetail.Doctor.BoardCertified = Board Certified`
4. In the **Project** window, navigate to **configuration→config→Page Configuration→pcf→addressbook**. Then double-click **AddressBookDoctorAdditionalInfoInputSet.Doctor** to open it in the editor.
5. Select the Input widget **Medical Specialty** and copy it, and then paste the copy below this widget.
6. Right-click the new field and choose **Change element type**.
7. Click **Boolean Radio Button Input** in the drop-down list.
8. Click the new **BooleanRadioInput** widget and set the following properties:

editable	true
id	DoctorBoardCertified
label	displaykey.Web.ContactDetail.Doctor.BoardCertified
required	false
value	(personVendor as Doctor).BoardCertified_Ext

Next steps

“Add the BoardCertified field to the claim contacts user interface” on page 154

Add the BoardCertified field to the claim contacts user interface

Enable the ClaimCenter user to edit the **BoardCertified** field of a **Doctor** entity when working with contacts for a claim.

Before you begin

Complete “Add the BoardCertified field to the Address Book user interface” on page 154.

Procedure

1. In the Guidewire Studio™ for ClaimCenter **Project** window, navigate to **configuration→config→Page Configuration→pcf→shared→contacts**. Then double-click `DoctorAdditionalInfoInputSet.Doctor` to open it in the editor.
2. Select the Input widget `Medical Specialty` and copy it, and then paste the copy below this widget.
3. Right-click the new field and choose **Change element type**.
4. Click **Boolean Radio Button Input** in the drop-down list.
5. Click the new `BooleanRadioInput` widget and set the following properties:

editable	true
id	DoctorBoardCertified
label	displaykey.Web.ContactDetail.Doctor.BoardCertified
required	false
value	Doctor.BoardCertified_Ext

Next steps

“Add a field to the `ABDoctor` subtype in `ContactManager`” on page 155

Add a field to the `ABDoctor` subtype in `ContactManager`

In this example of adding a field to a contact subtype, you extend the `ABDoctor` entity in `ContactManager`.

Before you begin

Complete “Add the `BoardCertified` field to the claim contacts user interface” on page 154.

Procedure

1. Start Guidewire Studio™ for `ContactManager`.
At a command prompt, navigate to the `ContactManager` installation folder and enter:

```
gwb studio
```

2. In the **Project** window, navigate to **configuration→config→Extensions→Entity**.
3. Double-click `ABDoctor.eti` to open it in the Entity editor.
4. Click **subtype** at the top of the **Element** hierarchy.
5. Click the drop-down list next to **+** and choose **column**.
6. Enter the following values for the new column:

Name	Value
name	BoardCertified_Ext
type	bit
nullok	true
desc	Is the doctor Board certified in the specialty?
default	false

7. Regenerate the *Data Dictionary* to ensure that your changes are correct.
At a command prompt, navigate to the `ContactManager` installation folder and enter:

```
gwb genDataDictionary
```

Next steps

“Add the new field to the ContactMapper class in ContactManager” on page 156

Add the new field to the ContactMapper class in ContactManager

To be able to send a new `ABContact` field to a Guidewire core application and receive updates for the field, you must add it to the `ContactMapper` class.

Before you begin

Complete “Add a field to the ABDoctor subtype in ContactManager ” on page 155.

About this task

For more information on the `ContactMapper` class, see “ContactMapper class” on page 275.

Procedure

1. In Guidewire Studio™ for ContactManager, click **Ctrl+N** and enter `ContactMapper`, and then double-click the `ab1000` version of the class in the search results to open it in the Gosu editor.
2. Press **Ctrl+F** and find the following line of code:

```
fieldMapping(ABDoctor#MedicalLicense),
```

3. Add the following code for the new field after the line for the `MedicalLicense` field:
`fieldMapping(ABDoctor#BoardCertified_Ext),`
4. If necessary, stop ContactManager as follows:
 - a. Open a command prompt in the ContactManager installation folder and then enter the following command:

```
gwb stopServer
```

- b. Wait to restart the ContactManager application and pick up the data model changes until after making the changes in the next step.

Next steps

“Add the BoardCertified field to the ContactManager user interface” on page 156

Add the BoardCertified field to the ContactManager user interface

Enable ContactManager users to use the new `BoardCertified` field when they create and edit `ABDoctor` contacts.

Before you begin

Complete “Add the new field to the ContactMapper class in ContactManager” on page 156.

About this task

You add the new field to the input set for the `ABDoctor` subtype for use in the `ContactBasicsDV.ABPerson` detail view.

Procedure

1. Navigate in the **Project** window to **configuration→config→Localizations→Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor.
2. Click **Ctrl+F** and find the following entry in the file:

```
Web.ContactDetail.Doctor.MedicalLicense
```

3. Insert a line after that entry and then enter the following key and value:
`Web.ContactDetail.Doctor.BoardCertified = Board Certified`
4. In the **Project** window, navigate to **configuration→config→Page Configuration→pcf→contacts→basics**.
5. Double-click `ABPersonVendorSpecialtyInputSet.ABDoctor` to open it in the editor.
6. Select the Input widget `Medical Specialty` and copy it, and then paste the copy below the `Medical Specialty` widget.
7. Right-click the new field and choose **Change element type**.
8. Click **Boolean Radio Button Input**.
9. Click the new `BooleanRadioInput` widget and set the following properties:

editable	true
id	DoctorBoardCertified
label	displaykey.Web.ContactDetail.Doctor.BoardCertified
required	false
value	(person as ABDoctor).BoardCertified_Ext

Next steps

“Restart both applications and test the new field” on page 157

Restart both applications and test the new field

In this step of adding a field to a contact subtype, you restart ClaimCenter and ContactManager to pick up all the changes made in the previous steps. Then you test the new field to ensure that both applications can use it and can communicate with each other.

Before you begin

Complete “Add the BoardCertified field to the ContactManager user interface” on page 156 before starting this step

Procedure

1. If necessary, stop both ClaimCenter and ContactManager, and then restart both applications.
2. Log in to ClaimCenter as a user who can create new contacts.
For example, log in as user `ssmith` with password `gw`.
3. Open an existing claim.
4. Click **Parties Involved** in the left Sidebar, and then on the **Contacts** screen choose **New Contact→Vendor→Doctor** to open the **New Doctor** screen.
5. Verify that **Board Certified** is listed in the **Additional Info** section.
6. Enter enough information to create a Doctor contact.
The minimum is **First name**, **Last name**, and **Tax ID**. Include a **Medical Specialty** and click **Yes** for **Board Certified**.
7. At the top of the edit screen in the **Roles** list view table, click **Add**.
8. Click the **Role** cell for the new entry and choose a role from the drop-down list that the new Doctor vendor will take on the claim. For example, choose **Doctor**.
9. Click **Update** to save the new contact to the claim.
ClaimCenter adds the new Doctor contact to the list of contacts on the **Contacts** screen. ClaimCenter also ensures that the Claim Party and Vendor tags are set for the contact and sends it to ContactManager.
10. Select the new doctor vendor in the **Contacts** list view table.
If you logged in as a user with permission to create contacts, below, on the **Basics** card, the message above the contact says:

This contact is linked to the Address Book and is in sync

This message means that the new contact has been saved to ContactManager, and the contact data for this contact on this claim is the same for ClaimCenter and ContactManager.

It is possible that the contact is still being saved to ContactManager and ClaimCenter has not yet been notified. In that case, the message you see is **Waiting for link message from ContactManager**. Refresh screen to get updated status. You can refresh the screen by clicking another contact or screen and then clicking this contact again.

11. Click **View in Address Book** to see a screen showing the data saved for this contact in ContactManager.
12. On this screen, click **Edit in ContactManager** to open ContactManager and edit the contact.
You might have to log in to ContactManager if your ClaimCenter user name is not in ContactManager. Log in as a ContactManager user who has ABContact view, edit, update, and delete permissions, such as the sample user aaggregate.
13. Click **Edit** and make some changes to the contact, such as a new address. Change the setting for the **Board Certified** field. Click **Update** to save your changes.
14. In ClaimCenter, click the **Address Book** tab.
15. On the **Search Address Book** screen, pick **Doctor** from the **Type** drop-down list and search for the doctor you added.
16. Click the contact found by **Search** and verify that the entry found by ClaimCenter in ContactManager is correct and that the data matches the changes you made in ContactManager.
17. Click the drop-down list for the **Claim** tab and choose the claim you previously edited.
18. Click **Parties Involved** in the left Sidebar, and then, on the **Contacts** screen, select the doctor you added to this claim.

Below, on the **Basics** card, the message above the contact says:

This contact is linked to the Address Book but is out of sync

This message means that the contact data you changed in ContactManager is now different from the contact data for this contact on this claim.

19. Click **Copy from Address Book** to update the contact data for this claim.

You see the changes on the **Basics** card and the following message:

This contact is linked to the Address Book and is in sync

20. Click **Edit** and change the setting for the **Board Certified** field, and then click **Update**.

Because you are logged in as a user who can edit contacts, this change is sent to ContactManager.

The message above the contact changes to:

This contact is linked to the Address Book but is out of sync

This message means that the change you made to the contact has not yet registered in ContactManager. It can take a few seconds for the message sent to ContactManager to take effect.

21. Click another contact in the **Contacts** list view table, and then click your original Doctor contact to refresh the **Basics** card.

If the message has not changed to say that the contact is in sync, wait a few seconds, and then click another contact and then this one again.

Eventually, you see the following message:

This contact is linked to the Address Book and is in sync

Adding a vendor contact subtype

You can extend contacts with a new subtype. In this multi-topic example, you add a new **Interpreter** subtype to ClaimCenter and its counterpart, **ABInterpreter**, to ContactManager. You then update the necessary files and screens to make the new field usable across the applications.

Add a new Contact subtype to ClaimCenter

Part of adding a vendor contact subtype is extending the **Contact** entity in ClaimCenter.

Before you begin

If you have not done so already:

- Follow the instructions for installing ContactManager in “Installing ContactManager” on page 55.
- You must have integrated ContactManager with ClaimCenter as described in “Integrating ContactManager with Guidewire core applications” on page 59.
- After installing ContactManager, for testing purposes, it would be helpful to import sample data as described in “Load sample data for ContactManager” on page 56.

About this task



This step is the first in the example “Adding a vendor contact subtype” on page 158. In this step, you add a new Interpreter entity in ClaimCenter.

Procedure


1. Start Guidewire Studio™ for ClaimCenter.

At a command prompt, navigate to the ClaimCenter installation folder and enter:

```
gwb studio
```

2. In the **Project** window, navigate to **configuration→config→Extensions→Entity**.
3. Right-click **Entity** and choose **New→Entity**.
4. In the **Entity** field, enter the following name:
Interpreter
5. Click the **Entity Type** drop-down list and choose **subtype**.
6. In the **Desc** field, enter the following description:
Interpreter
7. Click the **Supertype** search button  and then choose **PersonVendor** from the drop-down list.
8. Click **OK**.
9. In the Entity editor for the new entity, click **subtype** at the top of the **Element** hierarchy.
10. For **displayName**, enter Interpreter.
11. Click the drop-down list next to  and choose **column**.
12. Enter the following values for the new column:

Name	Value
name	InterpreterSpecialty
type	varchar
nullok	false
desc	Interpreter's language specialties

13. Click the drop-down list next to  and choose **columnParam**.
14. Enter the following values to specify the size of this varchar column:

Name	Value
name	size
value	30

15. Regenerate the *Data Dictionary*.

At a command prompt, navigate to the ClaimCenter installation folder and enter:

```
gwb genDataDictionary
```

16. Verify in the *Data Dictionary* that the data model extension is correct.
17. Close Guidewire Studio for ClaimCenter and then restart it to enable Studio to pick up the data model change.

Next steps

“Add the new subtype to the ContactMapper class in ClaimCenter” on page 160

Add the new subtype to the ContactMapper class in ClaimCenter

To be able to send a new Contact subtype to ContactManager and receive updates for it, you must add it to the ContactMapper class.

Before you begin

Complete “Add a new Contact subtype to ClaimCenter ” on page 158.

About this task

For more information on the ContactMapper class, see “ContactMapper class” on page 275.

Procedure

1. In Guidewire Studio™ for ClaimCenter, press Ctrl+N and enter ContactMapper, and then double-click the ab1000 version of the class in the search results to open it in the Gosu editor.
2. Press Ctrl+F and search for the following line of code:

```
fieldMapping(MedicalCareOrg#MedicalOrgSpecialty),
```

3. After the line for MedicalCareOrg, add the following code for the new entity:

```
fieldMapping(Interpreter#InterpreterSpecialty),
```

Note: You do not have to put this fieldMapping method call in this exact location. It must be in the method override `property get Mappings() : Set<CCPropertyMapping>`. Additionally, it is useful to group it with the other method calls after the comment `//Other Contact subtypes`.

Next steps

“Map the new subtype names in ClaimCenter” on page 160

Map the new subtype names in ClaimCenter

To be able to send a new Contact subtype to ContactManager and receive updates for it, you must map the ClaimCenter subtype name to the ContactManager ABContact subtype name. You do this mapping only in ClaimCenter in the class `gw.contactmapper.ab1000.CCNameMapper`.

Before you begin

Complete “Add the new subtype to the ContactMapper class in ClaimCenter” on page 160.

About this task

The ClaimCenter subtype is Interpreter and the ContactManager subtype, which you will create later, is ABInterpreter.

Procedure

1. In Guidewire Studio™ for ClaimCenter, press Ctrl+N and enter CCNameMapper.
2. In the search results, double-click the ab1000 version of the class to open it in the Gosu editor.
3. In the editor, press Ctrl+F and search for MedicalCareOrg. The search finds the following line of code:


```
.entity(MedicalCareOrg, "ABMedicalCareOrg")
```

4. Insert the following entry below the line for `MedicalCareOrg`.
`.entity(Interpreter, "ABInterpreter")`
5. If `ClaimCenter` is running, stop `ClaimCenter` as follows:
 - a. Open a command prompt in the `ClaimCenter` installation folder and then enter the following command:

```
gwb stopServer
```

- b. Do not restart the `ClaimCenter` application until after modifying the `ClaimCenter` Address Book in a later step.

Next steps

“Add display keys for the new subtype to `ClaimCenter`” on page 161

Add display keys for the new subtype to `ClaimCenter`

Adding a new `Interpreter` subtype requires changes to the user interface. Add display keys to `ClaimCenter` for menu items and fields.

Before you begin

Complete “Map the new subtype names in `ClaimCenter`” on page 160.

About this task

The new menu items and inputs for `Interpreter` have labels that require display keys, which support localization.

Procedure

1. Navigate in the **Project** window to **configuration→config→Localizations→Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor.
2. In the editor, press **Ctrl+F** and find the entry `Web.NewContactMenu.Doctor` in the file.
3. Insert a line under that entry, and then enter the following key and value:
`Web.NewContactMenu.Interpreter = Interpreter`
4. Use the search field at the top of the editor to find the entry `Web.ContactDetail.Email.Secondary` in the file.
5. Insert a new line under that entry, and then enter the following keys and values:
`Web.ContactDetail.Interpreter = Interpreter`
`Web.ContactDetail.Interpreter.InterpreterSpecialty = Interpreter Specialty`

Next steps

“Add the subtype to the `ClaimCenter` New Contact menu” on page 161

Add the subtype to the `ClaimCenter` New Contact menu

Modify the `ClaimCenter` New Contact menu to enable the user to create contacts with the new `Interpreter` subtype.

Before you begin

Complete “Add display keys for the new subtype to `ClaimCenter`” on page 161.

About this task

Note: In general, when you create new subtypes, the PCF files you need to edit depend on the parent of the subtype. For example, Person subtypes use a different set of PCF files from Company subtypes. You can discover the specific PCF files you need by examining those used by other subtypes. Additionally, the PCF files can use different modes to show appropriate fields for different subtypes.

Procedure

1. In Guidewire Studio™ for ClaimCenter, navigate in the **Project** window to **configuration→config→Page Configuration→pcf→claim→partiesinvolved**.
2. Double-click **ClaimContacts** to open that PCF file in the editor.
3. On the toolbar above the list view panel with the ID **PeopleInvolvedDetailedLV**, there is a **New Contact** button. Click the drop-down control to open the embedded menu.
4. Select the **Vendor** submenu, which has the **id** value **ClaimContacts_NewVendor**.
This submenu has menu elements for vendors like **Autobody Repair Shop**, **Doctor**, and **Medical Care Organization**.
5. Right-click the menu item widget for **Medical Care Organization** and copy it.
6. Paste the copy below the menu item for **Medical Care Organization**.
The new menu item is in the **Vendor** submenu. The new item turns red and stays that way until you enter the properties in the next step.
7. Click the new menu item widget and set the following properties:

action	<code>NewPartyInvolvedPopup.push(claim, typekey.Contact.TC_INTERPRETER)</code>
id	<code>ClaimContacts_Interpreter</code>
label	<code>displaykey.Web.NewContactMenu.Interpreter</code>
showConfirmMessage	<code>true</code>

8. Navigate in the **Project** window to **configuration→config→Page Configuration→pcf→shared→contacts**.
9. Double-click **ClaimNewServiceRequestSpecialistPickerMenuItemSet** to open it in the editor.
10. Right-click the menu item widget for **Medical Care Organization** and copy it.
11. Paste the copy below the menu item for **Medical Care Organization**.
The new menu item is in the **New Vendor** submenu. The new item turns red and stays that way until you enter the properties in the next step.
12. Click the new menu item widget and set the following properties:

action	<code>NewContactPopup.push(typekey.Contact.TC_INTERPRETER, parentContact, claim)</code>
id	<code>NewInterpreter</code>
label	<code>displaykey.Web.NewContactMenu.Interpreter</code>
showConfirmMessage	<code>true</code>

Next steps

“Add the new subtype to the Address Book input sets” on page 162

Add the new subtype to the Address Book input sets

Enable a ClaimCenter user to select an **Interpreter** contact and see the **Interpreter Specialty** field.

Before you begin

Complete “Add the subtype to the ClaimCenter New Contact menu” on page 161.

About this task

In this step you update the input sets used by the ClaimCenter Address Book detail view.

Procedure

1. In Guidewire Studio™ for ClaimCenter, navigate in the **Project** window to **configuration→config→Page Configuration→pcf→addressbook**.
2. Right-click `AddressBookDoctorAdditionalInfoInputSet.Doctor` and click **Copy**.
3. Right-click again and choose **Paste**.
4. In the **Copy** dialog box, enter the new name `AddressBookInterpreterAdditionalInfoInputSet.Interpreter.pcf`, and then click **OK**.
The new widget opens in the editor.
5. Click the name of the input set window, `AddressBookInterpreterAdditionalInfoInputSet`, to open the **Properties** tab.
6. Set the value of the mode property to `Interpreter`.
7. In the editor, select the **Medical License** input widget and set the following properties:

editable	true
id	InterpreterSpecialty
label	displaykey.Web.ContactDetail.Interpreter.InterpreterSpecialty
required	false
value	(personVendor as Interpreter).InterpreterSpecialty

8. Select the **Medical Specialty** widget and delete it.
9. If it exists, select the **Board Certified** widget and delete it.
10. In the **Project** window, navigate to **configuration→config→Page Configuration→pcf→addressbook**.
11. Open `AddressBookContactBasicsDV.Person` in the editor and click its name at the top to open the **Properties** tab.
12. For the mode property, add the `Interpreter` mode to the list:

```
Person|PersonVendor|Adjudicator|UserContact|Doctor|Attorney|Interpreter
```

13. Click **OK**.
14. Open `AddressBookAdditionalInfoInputSet.PersonVendor` in the editor and click its name at the top to open the **Properties** tab.
15. For the mode property, add the `Interpreter` mode to the list:

```
PersonVendor|Attorney|Doctor|Interpreter
```

16. Drag a new `InputSetRef` widget from the **Toolbox** and drop it below the second `InputSetRef` widget, the one containing the input set `AddressBookAttorneyAdditionalInfoInputSet`.
17. Click the new `InputSetRef` widget and set the following properties:

def	AddressBookInterpreterAdditionalInfoInputSet(contact as PersonVendor)
id	AddressBookInterpreterAdditionalInfoInputSet
mode	contact typeis Interpreter ? "Interpreter" : null

The mode statement enables the `Interpreter Specialty` field to display in the **Additional Info** section when the contact type is `Interpreter`.

Next steps

“Add the new contact to the shared contacts detail view” on page 164

Add the new contact to the shared contacts detail view

Enable the ClaimCenter user to create an Interpreter contact on the **New Contact** screen and see the **Interpreter Specialty** field.

Before you begin

Complete “Add the new subtype to the Address Book input sets” on page 162.

About this task

In this step, you update the input sets used by the shared contacts detail view.

Procedure

1. In Guidewire Studio™ for ClaimCenter, navigate in the **Project** window to **configuration→config→Page Configuration→pcf→shared→contacts**.
2. Open ContactBasicsDV.Person and click the top line in the screen, Detail View: ContactBasicsDV, to open the **Properties** tab.
3. For the mode property, add Interpreter to the list of modes, as follows:

```
Person|PersonVendor|Adjudicator|UserContact|Doctor|Attorney|Interpreter
```

4. In the same **Project** window folder, navigate to **pcf→shared→contacts**.
5. Right-click DoctorAdditionalInfoInputSet.Doctor and click **Copy**.
6. Right-click again and choose **Paste**.
7. In the **Copy** dialog box, enter the new name InterpreterAdditionalInfoInputSet.Interpreter.pcf, and then click **OK**.
The new widget opens in the editor.
8. In the **Project** window, open InterpreterAdditionalInfoInputSet.Interpreter.
9. In the editor, select the new widget by clicking its identifier at the top of the widget: InputSet: InterpreterAdditionalInfoInputSet to open the **Properties** tab.
10. Verify that Interpreter is set in the mode property.
11. Click the **Code** tab and replace the existing code with the following Gosu statement:
property get Interpreter() : Interpreter {return contactHandle.Contact as Interpreter;}
12. Select the Medical License input widget and change it to an Interpreter Specialty widget by setting the following properties:

editable	true
id	InterpreterSpecialty
label	displaykey.Web.ContactDetail.Interpreter.InterpreterSpecialty
required	false
value	Interpreter.InterpreterSpecialty

13. Select the Medical Specialty widget and delete it.
14. If it exists, select the Board Certified widget and delete it.
15. In the same **Project** window location, **pcf→shared→contacts**, open AdditionalInfoInputSet.PersonVendor.
16. In the editor, select the new widget by clicking its identifier at the top of the widget: InputSet: AdditionalInfoInputSet to open the **Properties** tab.
17. Click the mode property and add the Interpreter mode to the list:

```
PersonVendor|Attorney|Doctor|Interpreter
```

18. Drag a new InputSetRef widget from the **Toolbox** and drop it below the widget containing the input set AttorneyAdditionalInfoInputSet.
19. Click the new InputSetRef widget and set the following properties:

```
def InterpreterAdditionalInfoInputSet(contactHandle)
mode PersonVendor typeis Interpreter ? "Interpreter" : null
```

The mode statement enables the Interpreter Specialty field to show in the **Additional Info** section when the contact type is Interpreter.

Next steps

“Add a new ABContact subtype to ContactManager” on page 165

Add a new ABContact subtype to ContactManager

Part of adding a vendor contact subtype is extending the ABContact entity in ContactManager.

Before you begin

Complete “Add the new contact to the shared contacts detail view” on page 164.



About this task

In this step, you add an ABInterpreter subtype to ContactManager. This subtype is the counterpart of the Interpreter subtype you added to ClaimCenter in “Add a field to the Doctor subtype in ClaimCenter ” on page 152.

Procedure


1. Start Guidewire Studio™ for ContactManager. At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb studio
```


2. In the **Project** window open in **Project** view, navigate to **configuration→config→Extensions→Entity**.
3. Right-click **Entity** and choose **New→Entity**.
4. In the **Entity** field, type the following name: ABInterpreter
5. Click the **Entity Type** drop-down list and choose **subtype**.
6. In the **Desc** field, type the following description: Interpreter
7. Click the **Supertype** search button  and choose ABPersonVendor from the drop-down list.
8. Click **OK**.
9. In the Entity editor for the new entity, click **subtype** at the top of the **Element** hierarchy.
10. For **Display name**, enter Interpreter.
11. Click the drop-down list next to  and choose **column**.
12. Enter the following values for the new column:

Name	Value
name	InterpreterSpecialty
type	varchar
nullok	false

Name	Value
desc	Interpreter's language specialties

13. Click the drop-down list next to  and choose **columnParam**.
14. Enter the following values to specify the size of this varchar column:

Name	Value
name	size
value	30

15. Click **subtype** at the top of the **Element** hierarchy.
16. Click the drop-down list next to  and choose **index**.
17. Enter the following values for the new index:

Name	Value
name	intrprtrindx1
desc	Speed up searches using InterpreterSpecialty field
unique	true

18. Right-click **index** in the **Element** column, and choose **Add new→indexcol**.
This index column is the first of three to add to the index.
19. Enter the following values for the new **indexcol**:

Name	Value
name	InterpreterSpecialty
keyposition	1

20. Right-click **index** in the **Element** column, and choose **Add new→indexcol**.
21. Enter the following values for the new **indexcol**:

Name	Value
name	Retired
keyposition	2

22. Right-click **index** in the **Element** column, and choose **Add new→indexcol**.
23. Enter the following values for the new **indexcol**:

Name	Value
name	Id
keyposition	3

24. Regenerate the data dictionary to ensure that your changes are correct.
At a command prompt, navigate to the ContactManager installation folder and enter:

```
gwb genDataDictionary
```

Next steps

“Add the new subtype to the ContactMapper class in ContactManager” on page 167

Add the new subtype to the ContactMapper class in ContactManager

To be able to send the new subtype to a Guidewire core application and receive updates for the subtype, you must add it to the ContactMapper class.

Before you begin

Complete “Add a new ABContact subtype to ContactManager” on page 165.

About this task

For more information on the ContactMapper class, see “ContactMapper class” on page 275.

Procedure

1. In Guidewire Studio™ for ContactManager, press Ctrl+N and enter ContactMapper, and then double-click the ab1000 version of the class in the search results to open it in the Gosu editor.
2. Press Ctrl+F and search for the following line of code:

```
fieldMapping(ABMedicalCareOrg#MedicalOrgSpecialty),
```

3. After the line for ABMedicalCareOrg, add the following code for the new entity:

```
fieldMapping(ABInterpreter#InterpreterSpecialty),
```

Note: You do not have to put this fieldMapping method call in this exact location. It must be in the method with declaration override `property get Mappings() :`

`Set<PropertyMapping>`. Additionally, it is useful to group it with the other method calls after the comment `// Other ABContact subtypes`.

4. If ContactManager is running, you must stop and restart it to pick up the data model change. Wait to restart the ContactManager application until you complete the step “Restart both applications and test the new subtypes” on page 170.

Next steps

“Add display keys for the new subtype to ClaimCenter” on page 161

Add display keys for the new subtype to ContactManager

Adding a new ABInterpreter subtype requires changes to the user interface. Add display keys to ContactManager for menu items and fields.

Before you begin

Complete “Add the new contact to the shared contacts detail view” on page 164.

About this task

In this step, you add display keys to ContactManager to support changes to the user interface screens for the new ABInterpreter subtype.

Procedure

1. Open Guidewire Studio™ for ContactManager.
2. Navigate in the **Project** window to **configuration→config→Localizations→Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor.
3. Press Ctrl+F and search for `Web.NewContactMenu.Doctor`.
4. Add a new line under that entry, and then enter the following key and value:
`Web.NewContactMenu.Interpreter = Interpreter`
5. Use the Search field at the top of the editor to find the entry `Web.ContactDetail.Email.Secondary`.

6. Insert a new line under that entry, and then enter the following keys and values:


```
Web.ContactDetail.Interpreter = Interpreter
Web.ContactDetail.Interpreter.InterpreterSpecialty = Interpreter Specialty
```

Next steps

“Add the new subtype to the ContactManager actions menu” on page 168

Add the new subtype to the ContactManager actions menu

As part of adding a new vendor contact subtype, you can add an **Actions** menu entry for creating a new ABInterpreter contact.

Before you begin

Complete “Add display keys for the new subtype to ContactManager” on page 167.

Procedure

1. In Guidewire Studio™ for ContactManager, navigate in the **Project** window to **configuration→config→Page Configuration→pcf→contacts** and double-click ContactsMenuActions to open this file in the PCF editor.
2. Drag a MenuItem widget from the **Toolbox** on the right and drop it under the Doctor menu item.
The new menu item is in the Vendor submenu, which has ID ContactsMenuActions_PersonVendorMenuItem.
3. Click the new MenuItem and enter the following values for the **Basic properties** of this Interpreter menu item:

action	NewContact.go(entity.ABInterpreter)
id	ContactsMenuActions_InterpreterMenuItem
label	displaykey.Web.NewContactMenu.Interpreter
showConfirmMessage	true

Next steps

“Add ABInterpreter fields to ContactManager modal input sets” on page 168

Add ABInterpreter fields to ContactManager modal input sets

Part of adding a new ABContact subtype is configuring a screen for entering data for the new entity.

Before you begin

Complete “Add the new subtype to the ContactManager actions menu” on page 168.

About this task

In this step, you add a modal input set supporting the ABInterpreter contact.

Procedure

1. In Guidewire Studio™ for ContactManager, navigate in the **Project** window to **configuration→config→Page Configuration→pcf→contacts→basics**.
2. Right-click ABPersonVendorSpecialtyInputSet.ABDoctor and choose **Copy**.
3. Right-click again and choose **Paste**.
4. Name the new file ABPersonVendorSpecialtyInputSet.ABInterpreter.pcf and click **OK**.
5. In the new PCF file, click its name, ABPersonVendorSpecialtyInputSet, at the top to open the **Properties** tab.
6. Verify that the mode property is set to ABInterpreter.

7. In the editor, click the **Medical License** input widget and change it to an **Interpreter Specialty** widget by setting the following properties:

editable	true
id	InterpreterSpecialty
label	displaykey.Web.ContactDetail.Interpreter.InterpreterSpecialty
required	false
value	(person as ABInterpreter).InterpreterSpecialty

8. Select the **Medical Specialty** widget and delete it.
9. If it exists, select the **Board Certified** widget and delete it.
10. In the same location in the **Project** window, **pcf**→**contacts**→**basics**, open `ABPersonVendorInputSet.ABPersonVendor`.
11. Click the name of the PCF file at the top, `ABPersonVendorInputSet`, to open the **Properties** tab.
12. Add **ABInterpreter** to the mode property as follows:

```
ABPersonVendor|ABAttorney|ABDoctor|ABInterpreter
```

13. In the same location in the **Project** window, **pcf**→**contacts**→**basics**, open `ContactBasicsDV.ABPerson` and click its name at the top to open the **Properties** tab.
14. Add **ABInterpreter** to the mode property, as follows:

```
ABPerson|ABPersonVendor|ABAdjudicator|ABUserContact|ABDoctor|ABAttorney|ABPolicyPerson|ABInterpreter
```

15. Click **OK** to save the new mode.

Next steps

“Add the new subtype to the ContactManager contact picker menus” on page 169

Add the new subtype to the ContactManager contact picker menus

Part of adding an **ABContact** extension is configuring the **ContactManager** contact picker menu so the user can choose the new entity.

Before you begin

Complete “Add **ABInterpreter** fields to **ContactManager** modal input sets” on page 168.

About this task

Add the new **ABInterpreter** entity to the **ContactManager** picker menu.

Procedure

1. In Guidewire Studio™ for **ContactManager**, navigate in the **Project** window to **configuration**→**config**→**Page Configuration**→**pcf**→**contacts**.
2. Double-click `NewContactPickerMenuItemSet` to open this file in the PCF editor.
3. Select the submenu labeled **Vendor**, which has the following **id** property:

```
NewContactPickerMenuItemSet_PersonVendorMenuItem
```

4. Copy the **Doctor** menu item and paste the copy under the existing **Doctor** menu item.
5. Select the new menu item.
The menu item turns red, indicating an error. It stays red until you set the properties in the next step.
6. Set the following properties for this new menu item:

action	NewContactPopup.push(entity.ABInterpreter, parentContact)
id	NewContactPickerMenuItemSet_InterpreterMenuItem
label	displaykey.Web.NewContactMenu.Interpreter
showConfirmMessage	true
visible	requiredContactType.isAssignableFrom(entity.ABInterpreter)

Next steps

“Restart both applications and test the new subtypes” on page 170


Restart both applications and test the new subtypes

After adding new contact subtypes and configuring the user interfaces in ClaimCenter and ContactManager, restart both applications and test that they support and communicate about the new subtypes.

Before you begin

Complete “Add the new subtype to the ContactManager contact picker menus” on page 169.

Procedure

1. If they are running, stop the ClaimCenter and ContactManager servers, and then restart them to refresh each application with your PCF and data model changes.
2. Log in to ClaimCenter as a user who can create new contacts.
For example, log in as user `ssmith` with password `gw`.
3. Open a claim and navigate to the **Parties Involved**→**Contacts** screen.
4. Choose **New Contact**→**Vendor**→**Interpreter** to see the **New Interpreter** dialog. Verify that **Interpreter Specialty** is listed in the **Additional Info** section.
5. Enter enough information to create an interpreter contact, such as name, address, interpreter specialty, and tax ID.
6. Add a role for the contact on the claim, and then click **Update** to save the new contact information.
7. Because you are logged in as a user with permission to create contacts, ClaimCenter sends the new contact to ContactManager.
If you click the contact on the Contacts screen right away, you are likely to see the following message:
Waiting for link message from ContactManager. Refresh screen to get updated status.
8. You can refresh the screen by clicking another contact on the list and then clicking this one again.
Eventually, you see the following message:
This contact is linked to the Address Book and is in sync
9. Click the **Address Book** tab.
10. In the **Search Address Book** screen, pick **Interpreter** from the **Type** drop-down list and search for the new contact.
11. Select the new contact found by **Search** and verify that the entry is correct and that you can see the **Interpreter Specialty** field.
12. Return to the claim and click **Actions**→**New**→**Service**.
13. For **Request Type**, choose **Perform Service**.
14. For **Vendor Name**, click the contact picker and choose **New Vendor**→**Interpreter** from the drop-down list.
The *contact picker* is a second drop-down button to the right of the field:

15. Enter a name, an address, an interpreter specialty, and a tax ID, and then click **OK**.
16. On the **Create Service Requests** screen under **Services to Perform**, click **Add** to specify a service to perform.
There is no Interpreter service in the base configuration. Just pick any service on the list.

17. At the bottom of the screen, choose a **Service Address**, and then click **Submit**.
18. In the Sidebar, click **Parties Involved**.
The **Contacts** screen opens.
19. Verify that your new contacts have been added. Click one of the contacts to open the **Basics** card and verify that **Interpreter Specialty** is listed under **Additional Info**.
20. Log in to ContactManager as a user who can create new contacts.
For example, log in as user `aappleagate` with password `gw`.
21. In the Sidebar, choose **Search**, pick **Interpreter** as the **Contact Type**, and search for one of the interpreter contacts you created in ClaimCenter.
22. Click the contact's name to see the details screen for the contact.
23. On the **Basics** tab, click **Edit** and change the value of **Interpreter Specialty**, and then click **Update** to save the change.
24. In the Sidebar, choose **Actions**→**New Person**→**Vendor**→**Interpreter** to see the **New Interpreter** screen.
25. Verify that **Interpreter Specialty** is listed in the **Additional Info** section.
26. Enter enough information to create an Interpreter contact. For **Tags**, click **Vendor**. Click **Update** to save the new contact.
27. Return to ClaimCenter.
28. Click the **Address Book** tab.
29. On the **Search Address Book** screen, pick **Interpreter** from the drop-down list and search for the new contact you created in ContactManager.
30. Click the contact found by **Search** and verify that the entry is correct.
31. Search for the interpreter that you originally created in ClaimCenter and then edited in ContactManager.
32. Click that contact and verify that the changes you made in ContactManager are in this contact's data.

Extending contacts with an array

You can extend the `ABContact` entity in ContactManager and the `Contact` entity in ClaimCenter with an array. In this example, a multi-step process divided into a series of topics, the array is composed of states that comprise the service area for the contact. After creating and extending entities, you update the necessary files and screens to make the new array usable across the applications.

Create a service state entity in ContactManager

Part of extending contacts with an array is to create the entity that will be in the array.

About this task

This step is the first in the example described at “Extending contacts with an array” on page 171. In this step, you add an entity to ContactManager representing a state in which a contact provides service.

Procedure

1. Start Guidewire Studio™ for ContactManager.
At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb studio
```
2. In the **Project** window, navigate to **configuration**→**config**→**Extensions**→**Entity**, and then right-click **Entity** and choose **New**→**Entity**.
3. Enter the following file name: `ContactServiceState`
4. Enter the following for **Desc**:
Represents a state where the contact provides services
5. In addition to **Extendable** and **Final**, which are already selected, select the **Exportable** check box.

6. Click **OK**.
7. In the editor, click **entity** at the top of the **Element** hierarchy
8. Click the drop-down list for **+** and choose **implementsEntity**, and then choose **ABLinkable** from the drop-down list for the **name** value.
9. Click the drop-down list for **+** and choose **implementsEntity**, and then choose **Extractable** from the drop-down list for the **name** value.

Choosing **Extractable** makes the entity part of the **ABContact** entity graph. If you implement personal data destruction, being part of the entity graph makes it possible to destroy the entity if necessary. See “Extensions of **ABContact** and other entities and the domain graph” on page 230.

10. Click the drop-down list for **+** and choose **implementsEntity**, and then choose **Obfuscatable** from the drop-down list for the **name** value.

Choosing **Obfuscatable** makes it possible to obfuscate the entity as part of personal data destruction. See “Implementing the **Obfuscatable** delegate in an entity” on page 235.

11. Click the drop-down list for **+** and choose **implementsInterface**.

Enter the following values:

- For **iface**, enter `gw.api.obfuscation.Obfuscator`.
- For **impl**, enter `gw.personaldata.obfuscation.DefaultPersonalDataObfuscator`.

This step specifies how fields of the entity will be obfuscated. See:

- “Implementing the **Obfuscator** interface in an entity” on page 235
- “Personal data obfuscation class hierarchy” on page 236

12. Click **entity** at the top of the **Element** hierarchy.

13. Click the drop-down list for **+** and choose **foreignKey**, and then enter the following values:

Name	Value
name	Contact
fkentity	ABContact
nullok	false
columnName	ContactID
desc	Contact that this Service State row relates to

14. Click **entity** at the top of the **Element** hierarchy.

15. Click the drop-down list for **+** and choose **typekey**, and then enter the following values:

Name	Value
name	ServiceState
typelist	State
nullok	false
desc	State serviced by the contact
exportable	true (default value)
loadable	true (default value)

16. Click **entity** at the top of the **Element** hierarchy.

17. Click the drop-down list next to **+** and choose **index**.

18. Enter the following values for the new index:

Name	Value
name	absrvstatelinkid

Name	Value
desc	Preserve uniqueness of LinkID
unique	true

19. Right-click **index** in the **Element** column, and choose **Add new→indexcol**.

This index column is the first of two to add to the index.

20. Enter the following values for the new **indexcol**:

Name	Value
name	LinkID
keyposition	1

21. Right-click **index** in the **Element** column, and choose **Add new→indexcol**.

22. Enter the following values for the new **indexcol**:

Name	Value
name	Retired
keyposition	2

23. Click **entity** at the top of the **Element** hierarchy.

24. Click the drop-down list next to **+** and choose **index**.

25. Enter the following values for the new index:

Name	Value
name	ind1
unique	true

26. Right-click the **ind1 index** in the **Element** column, and choose **Add new→indexcol**.

This index column is the first of three to add to the index.

27. Enter the following values for the new **indexcol**:

Name	Value
name	ServiceState
keyposition	1

28. Right-click the **ind1 index** in the **Element** column, and choose **Add new→indexcol**.

29. Enter the following values for the new **indexcol**:

Name	Value
name	Retired
keyposition	2

30. Right-click the **ind1 index** in the **Element** column, and choose **Add new→indexcol**.

31. Enter the following values for the new **indexcol**:

Name	Value
name	ContactID
keyposition	3

32. Click **entity** at the top of the **Element** hierarchy.

33. Click the drop-down list next to **+** and choose **index**.

34. Enter the following values for the new index:

Name	Value
name	ind2
unique	true

35. Right-click the ind2 index in the **Element** column, and choose **Add new→indexcol**.

This index column is the first of three to add to the index.

36. Enter the following values for the new indexcol:

Name	Value
name	ContactID
keyposition	1

37. Right-click the ind2 index in the **Element** column, and choose **Add new→indexcol**.

38. Enter the following values for the new indexcol:

Name	Value
name	Retired
keyposition	2

39. Right-click the ind2 index in the **Element** column, and choose **Add new→indexcol**.

40. Enter the following values for the new indexcol:

Name	Value
name	ServiceState
keyposition	3

Next steps

“Add the new array to the ABContact entity” on page 174

Add the new array to the ABContact entity

Part of extending contacts with an array is to extend the ABContact entity in ContactManager to enable it to use an array of service state entities.

Before you begin

Complete “Create a service state entity in ContactManager” on page 171.

About this task

The array has the `triggersValidation` attribute set to `true` to enable triggering of validation rules when an element of the array changes.

Procedure

1. In Guidewire Studio™ for ContactManager, navigate in the **Project** window to **configuration→config→Extensions→Entity**, and then double-click `ABContact.etx` to open this file in the Entity editor.
2. In the editor, click **entity (extension)** at the top of the **Element** hierarchy
3. Click the drop-down list for **+** and choose **array**, and then enter the following values:

Name	Value
name	ContactServiceArea
arrayentity	ContactServiceState
arrayfield	Contact
desc	Geographical area where the contact provides service
triggersValidation	true

4. If necessary, stop ContactManager.

Open a command prompt in the ContactManager installation folder and then enter the following command:

```
gwb stopServer
```

5. Regenerate the *Data Dictionary* to ensure that the data model updates are correct.

At a command prompt, navigate to the ContactManager installation folder and enter:

```
gwb genDataDictionary
```

Next steps

“Map the entity and array extensions in ContactManager” on page 175

Map the entity and array extensions in ContactManager

After creating new ABContact entities and array extensions, you need to map them from ContactManager to any core applications that support them.

Before you begin

Complete “Add the new array to the ABContact entity” on page 174.

About this task

In this topic, you add the new array reference and the entity it references to the ContactMapper class in ContactManager. For more information on this class, see “ContactMapper class” on page 275.

Procedure

1. In Guidewire Studio™ for ContactManager, press Ctrl+N and search for the ContactMapper class.
2. Double-click the ab1000 version of the class in the search results to open it in the editor.
3. Find the Mappings method, which has the following declaration:

```
override property get Mappings() : Set<PropertyMapping>
```

4. At the end of the method, add a comma to the last line of code so you can add more code. For example:

```
fieldMapping(ABContactCategoryScore#Score),
```

5. After this last line of code, add an arrayMapping method for ABContact#ContactServiceArea, preceded with a comment for this part of the Mappings method:

```
//ContactServiceState mapping
arrayMapping(ABContact#ContactServiceArea),
```

6. Add fieldMapping methods for the elements of the ContactServiceState entity, which the ABContact array reference refers to:

```
fieldMapping(ContactServiceState#LinkID).withMappingDirection(TO_XML),
fieldMapping(ContactServiceState#External_PublicID),
```

```
fieldMapping(ContactServiceState#External_UniqueID),
fieldMapping(ContactServiceState#ServiceState)
```

Next steps

“Add support for service states to the ContactManager user interface” on page 176

Add support for service states to the ContactManager user interface

Enable ContactManager users to work with the ServiceState property when editing and viewing contacts.

Before you begin

Complete “Map the entity and array extensions in ContactManager” on page 175.

Procedure

1. Open Guidewire Studio™ for ContactManager.
2. Navigate in the **Project** window to **configuration→config→Localizations→Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor.
3. Press **Ctrl+F** and search for the entry `Web.ContactDetail.RetiredMessage` in the file.
4. Add a new line below that entry, and then enter the following display keys:


```
Web.ContactDetail.ServiceStateName = State Name
Web.ContactDetail.ServiceStates = Service States
```
5. In the **Project** window, navigate to **configuration→config→Page Configuration→pcf→contacts→basics**, and then double-click `ContactBasicsDV.ABCompany` to edit this PCF file.
6. On the right, at the bottom of the input column containing the `TextAreaInput` called `Notes`, drag a `ListViewInput` from the **Toolbox** and drop it above `Notes`.
7. Click the new `ListViewInput` and set the following properties:

label	<code>displaykey.Web.ContactDetail.ServiceStates</code>
labelAbove	<code>true</code>
boldLabel	<code>true</code>

8. From the **Toolbox**, drag a `RowIterator` and drop it on the new list view panel, and then click it and set the following properties:

editable	<code>true</code>
elementName	<code>currentServiceState</code>
toAdd	<code>contact.addToContactServiceArea(currentServiceState)</code>
toRemove	<code>contact.removeFromContactServiceArea(currentServiceState)</code>
value	<code>contact.ContactServiceArea</code>
canPick	<code>true</code>

The new list view input remains red until you add the Row widget in the next step.

9. From the **Toolbox**, drag a `Row` and drop it on the `RowIterator`, and then drag a `Cell`, and drop it on the `Row`.
10. Click the cell and set the following cell properties:

editable	<code>true</code>
id	<code>ServiceState</code>
label	<code>displaykey.Web.ContactDetail.ServiceStateName</code>

value	currentServiceState.ServiceState
-------	----------------------------------

unique	true
--------	------

- When you dropped the `RowIterator`, the PCF editor created a `ListViewPanel` as a container for it. Click the `ListViewPanel` and set its `id` property to `CurrentServiceStateLV`.
- From the **Toolbox** on the right, drag a `ToolBar` and drop it above the `ListViewPanel` named `CurrentServiceStateLV`.
- From the **Toolbox**, drag an `IteratorButtons` widget and drop in on the toolbar so you can add and remove states.
- Click the new `IteratorButtons` widget and set the property `iterator` to `CurrentServiceStateLV`.

Next steps

“Create a service state entity in ClaimCenter” on page 177

Create a service state entity in ClaimCenter

Part of extending contacts with an array is to create the `ClaimCenter` entity that will be in the array.

Before you begin

Complete “Add support for service states to the ContactManager user interface” on page 176.

About this task

Create an entity that is the `ClaimCenter` counterpart of the `ContactManager` `ContactServiceState` entity.

Procedure

- If necessary, start Guidewire Studio™ for ClaimCenter.
At a command prompt, navigate to the `ClaimCenter` installation folder and enter the following command:


```
gwb studio
```

- In the **Project** window, navigate to **configuration→config→Extensions→Entity**.
- Right-click **Entity** and choose **New→Entity**.
- Enter the following file name: `ContactServiceState`
- Enter the following for **Desc**:
Represents a state where the contact provides services
- In addition to **Extendable** and **Final**, select **Exportable**.
- Click **OK**.
- In the editor, click **entity** at the top of the **Element** hierarchy
- Click the drop-down list for **+** and choose **implementsEntity**, and then choose **Extractable** from the drop-down list for the **name** value.
- Repeat the two previous steps, but use them to create **implementsEntity** elements named **OverlapTable** and **AddressBookLinkable**.
- Click **entity** at the top of the **Element** hierarchy.
- Click the drop-down list next to **+** and choose **foreignKey**, and then enter the following values:


Name	Value
name	Contact
fkentity	Contact
nullok	false

Name	Value
columnName	ContactID
desc	Contact that this Service State row relates to

13. Click **entity** at the top of the **Element** hierarchy.

14. Click the drop-down list next to  and choose **typekey**, and then enter the following values:

Name	Value
name	ServiceState
nullok	false
typelist	State
desc	State serviced by the contact
exportable	true (default value)
loadable	true (default value)

15. Click **entity** at the top of the **Element** hierarchy, and then click the drop-down list next to  and choose **index**.

16. Enter the following values for the new index:

Name	Value
name	ind1
unique	true

17. Right-click the ind1 **index** in the **Element** column, and choose **Add new→indexcol**.

This index column is the first of three to add to the index.

18. Enter the following values for the new indexcol:

Name	Value
name	ServiceState
keyposition	1

19. Right-click the ind1 **index** in the **Element** column, and choose **Add new→indexcol**.

20. Enter the following values for the new indexcol:


Name	Value
name	Retired
keyposition	2

21. Right-click the ind1 **index** in the **Element** column, and choose **Add new→indexcol**.

22. Enter the following values for the new indexcol:

Name	Value
name	ContactID
keyposition	3

23. Click **entity** at the top of the **Element** hierarchy.

24. Click the drop-down list next to  and choose **index**.

25. Enter the following values for the new index:

Name	Value
name	ind2
unique	true

26. Right-click the ind2 index in the **Element** column, and choose **Add new→indexcol**.

This index column is the first of three to add to the index.

27. Enter the following values for the new indexcol:

Name	Value
name	ContactID
keyposition	1

28. Right-click the ind2 index in the **Element** column, and choose **Add new→indexcol**.

29. Enter the following values for the new indexcol:

Name	Value
name	Retired
keyposition	2

30. Right-click the ind2 index in the **Element** column, and choose **Add new→indexcol**.

31. Enter the following values for the new indexcol:

Name	Value
name	ServiceState
keyposition	3

Next steps

“Add the new array to the Contact entity in ClaimCenter” on page 179

Add the new array to the Contact entity in ClaimCenter

Extend the Contact entity in ClaimCenter to enable it to use an array of service state entities.

Before you begin

Complete “Create a service state entity in ClaimCenter” on page 177.

About this task

The array of service state entities has the `triggersValidation` attribute set to `true` to enable validation rules to be triggered when an element of the array changes. Additionally, to simplify communications between ContactManager and ClaimCenter, the array has the same name as the array added to ContactManager.

Procedure

1. In Guidewire Studio™ for ClaimCenter, navigate in the **Project** window to **configuration→config→Extensions→Entity**, and then double-click `Contact.etx` to open it in the editor.
2. In the editor, click **extension** at the top of the **Element** hierarchy
3. Click the drop-down list next to **+** and choose **array**, and then enter the following values:

Name	Value
name	ContactServiceArea

Name	Value
arrayentity	ContactServiceState
arrayfield	Contact
desc	Geographical area where the contact provides service
triggersValidation	true

4. If necessary, stop ClaimCenter.
Open a command prompt in the ClaimCenter installation folder and then enter the following command:

```
gwb stopServer
```

5. Regenerate the *Data Dictionary* to ensure that the data model updates are correct.
At a command prompt, navigate to the ClaimCenter installation folder and enter:

```
gwb genDataDictionary
```

Next steps

“Map the new entity and array extensions in ClaimCenter” on page 180

Map the new entity and array extensions in ClaimCenter

After creating new Contact entities and array extensions, you need to map them from ClaimCenter to ContactManager.

Before you begin

Complete “Add the new array to the Contact entity in ClaimCenter ” on page 179.

About this task

In this step, you add the new array reference and the entity it references to the `ContactMapper` class in ClaimCenter. For more information on this class, see “ContactMapper class” on page 275. Additionally, for more information on the last step in this topic, see “ContactManager link IDs and comparison to other IDs” on page 260.

Procedure

1. In Guidewire Studio™ for ClaimCenter, press **Ctrl+N** and search for the class `ContactMapper`, and then double-click the **ab1000** version of this class in the search results to open it in the editor.
2. Find the `Mappings` method, which has the following declaration:

```
override property get Mappings() : Set<CCPropertyMapping>
```

In this method, you will add a method for Contact array reference `ContactServiceArea` and methods that map the elements of the array, `ContactServiceState`.

3. At the end of the method, add a comma to the last line of code so you can add more code. For example:

```
.withEntityBeanBlock( \ lm, bp -> populateBeanFromXml(bp)),
```

4. After this last line of code, add an `arrayMapping` method for `Contact#ContactServiceArea`, preceded with a comment for this part of the `Mappings` method:

```
//ContactServiceState mapping  
arrayMapping(Contact#ContactServiceArea),
```

5. Add `fieldMapping` methods for the elements of the `ContactServiceState` entity, which the `ABContact` array reference refers to:

```
fieldMapping(ContactServiceState#AddressBookUID)
    .withABName(MappingConstants.LINK_ID),
fieldMapping(ContactServiceState#PublicID)
    .withMappingDirection(TO_XML)
    .withABName(MappingConstants.EXTERNAL_PUBLIC_ID),
fieldMapping(ContactServiceState#ServiceState)
```

This code maps between the following `ContactServiceState` fields in `ClaimCenter` and `ContactManager`.

ClaimCenter	ContactManager	Direction
AddressBookUID	LINK_ID	Both ways
PublicID	EXTERNAL_PUBLIC_ID	From ClaimCenter to ContactManager only
ServiceState	ServiceState	Both ways

Next steps

“Add support for service state searches to the Address Book” on page 181

Add support for service state searches to the Address Book

Enable `ClaimCenter` users to use the Address Book to search for contacts in `ContactManager` by `ServiceState`.

Before you begin

Complete “Map the new entity and array extensions in `ClaimCenter`” on page 180.

Procedure

1. Open Guidewire Studio™ for `ClaimCenter`.
2. Navigate in the **Project** window to **configuration**→**config**→**Localizations**→**Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor.
3. Press **Ctrl+N** and search for `Web.ContactDetail.RetiredMessage`.
4. Insert a new line below that one and add the following entries:


```
Web.ContactDetail.ServiceStateName = State Name
Web.ContactDetail.ServiceStates = Service States
```
5. In the **Project** window, navigate to **configuration**→**Page Configuration**→**pcf**→**addressbook**, and then right-click **addressbook** and click **New**→**PCF File**.
6. For the **File name**, enter `AddressBookServiceStates`.
7. For **File type**, click **List View**, and then click **OK** to edit this PCF file.
8. Click the new `ListViewPanel` to open its **Properties** window, and then click the **Required Variables** tab.
9. Click **+** and add the following new required variable:

name	contact
type	Contact

10. Click the **Exposes** tab, and then click **+** and choose **ExposerIterator**.
11. Enter the following values:

valueType	entity.ContactServiceState
widget	AddressBookServiceStatesLV

12. From the **Toolbox** on the right, drag a `RowIterator` and drop in on the list view.
13. Select the `RowIterator` widget and set the following properties:

editable	true
----------	------

elementName	currentServiceState
toAdd	contact.addToContactServiceArea(currentServiceState)
toRemove	contact.removeFromContactServiceArea(currentServiceState)
value	contact.ContactServiceArea
canPick	true

14. From the **Toolbox** on the right, drag a new Row and drop it on the RowIterator.
15. From the **Toolbox** on the right, drag a new Cell and drop it on the Row, and then set the following cell properties:

editable	true
id	ServiceState
label	displaykey.Web.ContactDetail.ServiceStateName
value	currentServiceState.ServiceState
unique	true

16. If the outline of the ListViewPanel remains red, it is likely that the editor has lost the value type of the **ExposerIterator** named AddressBookServiceStatesLV in the **Exposes** tab. To see if that is the case, click the ListViewPanel to open its **Properties** window. Then, as described in “Add support for service state searches to the Address Book” on page 181, click the **Exposes** tab, and then click AddressBookServiceStatesLV and, if necessary, reenter the following **valueType**:
entity.ContactServiceState
17. In the **Project** window, navigate to **configuration**→**config**→**Page Configuration**→**pcf**→**addressbook**, and then double-click AddressBookContactBasicsDV.Company to edit this PCF file.
18. On the right, in the InputColumn with the TextAreaInput called Notes at its bottom, drag a ListViewInput widget and drop it above Notes. Then enter the following attributes:

def	AddressBookServiceStatesLV(contact)
label	displaykey.Web.ContactDetail.ServiceStates
labelAbove	true
boldLabel	true

The new list view input remains red until you add the **Toolbar** widget in the next step.

19. From the **Toolbox** on the right, drag a Toolbar and drop it in the new ListViewInput above the ListViewPanel.
20. From the **Toolbox** on the right, drag an IteratorButtons widget and drop it on the toolbar so you can add and remove states.
21. Click the new IteratorButtons widget and set the **iterator** property value to AddressBookServiceStatesLV.AddressBookServiceStatesLV.

Next steps

“Enable addition of service states to a company on a claim” on page 182

Enable addition of service states to a company on a claim

As part of extending contacts with an array, extend the ClaimCenter user interface to enable users to add service states to a company that is associated with a claim.

Before you begin

Complete “Add support for service state searches to the Address Book” on page 181.

Procedure

1. In Guidewire Studio™ for ClaimCenter, navigate in the **Project** window to **configuration→config→Page Configuration→pcf→shared→contacts**.
2. Right-click **contacts** and click **New→PCF File**.
3. For the **File name**, enter `VendorServiceStates`, and for **File type** choose **List View**, and then click **OK** to edit this PCF file.
4. Click the new list view panel to open its **Properties** window, then click the **Required Variables** tab.
5. Click **+** and add the following new required variable:

name	<code>contactHandle</code>
type	<code>contact.ContactHandle</code>

6. Click the **Code** tab and enter the following code:

```
property get contact() : Contact { return contactHandle.Contact }
```
7. Click the **Exposes** panel, and then click **+** and click **ExposerIterator**.
8. Enter the following values:

valueType	<code>entity.ContactServiceState</code>
widget	<code>VendorServiceStatesLV</code>

9. From the **Toolbox** on the right, drag a **RowIterator** and drop it on the new list view panel, and then set the following properties:

editable	<code>true</code>
elementName	<code>currentServiceState</code>
toAdd	<code>contact.addToContactServiceArea(currentServiceState)</code>
toRemove	<code>contact.removeFromContactServiceArea(currentServiceState)</code>
value	<code>contact.ContactServiceArea</code>
canPick	<code>true</code>

10. From the **Toolbox**, drag a new **Row** and drop it on the **RowIterator**.
11. From the **Toolbox**, drag a new **Cell**, drop it on the **Row**, and then set the following cell properties:

editable	<code>true</code>
id	<code>ServiceState</code>
label	<code>displaykey.Web.ContactDetail.ServiceStateName</code>
value	<code>currentServiceState.ServiceState</code>
unique	<code>true</code>

12. If the outline of the `ListViewPanel` remains red, it is likely that the editor has lost the `valueType` of the **ExposeIterator** in the **Exposes** tab, `VendorServiceStatesLV`. To see if that is the case, click the `ListViewPanel` to open its **Properties** window. Then, click the **Exposes** tab, click `VendorServiceStatesLV`, and, if necessary, reenter the following `valueType`:
`entity.ContactServiceState`
13. In the **Project** window, navigate to **configuration→config→Page Configuration→pcf→shared→contacts**, and then double-click `ContactBasicsDV.Company` to edit this PCF file.
14. On the right, in the input column containing the `TextAreaInput` called `Notes`, drag a `ListViewInput` from the **Toolbox** and drop it above `Notes`.
15. Click the new `ListViewInput` widget and set the following properties:

def	VendorServiceStatesLV(contactHandle)
label	displaykey.Web.ContactDetail.ServiceStates
labelAbove	true
boldLabel	true

The new list view input remains red until you add the **Toolbar** widget in the next step.

16. From the **Toolbox** on the right, drag a **Toolbar** and drop it in the **ListViewInput**.
17. From the **Toolbox** on the right, drag an **IteratorButtons** widget and drop it on the toolbar so you can add and remove states.
18. Click the new **IteratorButtons** widget and set the following property:

iterator	VendorServiceStatesLV.VendorServiceStatesLV
-----------------	---

Next steps

“Test service state entity and array extensions” on page 184

Test service state entity and array extensions

After extending contacts with an array and configuring the **ContactManager** and **ClaimCenter** user interfaces, test that the extensions work in both applications.

Before you begin

Complete “Enable addition of service states to a company on a claim” on page 182.

Procedure

1. Shut down **ClaimCenter** and **ContactManager** if they are running, and then restart both applications.
2. Log in to **ClaimCenter** as a user who can create new contacts.
For example, if you have loaded sample data, log in as user `ssmith` with password `gw`.
3. Click the **Claim** tab and open an existing claim.
4. Click **Parties Involved** in the Sidebar, and then on the **Contacts** screen click **New Contact→Company**.
5. On the **New Company** screen, **Service States** is listed above the **Notes** section on the right.
6. Click **Add** to add a row for a state, and then click the new **State Name** field and select a state from the list.
7. Select the new row, and then click **Remove** to delete it.
8. Add several states, and then enter enough information to create the new company.
9. Under **Roles** near the top of the screen, click **Add**, and then add a role for this company for the current claim, such as **Vendor**.
10. Click **Update** to add the company to the list on the **Contacts** screen.
11. Click the company in the list on the **Contacts** screen.
On the **Basics** card, you see the following message:
This contact is linked to the Address Book and is in sync
Because you logged in as a user with permission to create and edit contacts, **ClaimCenter** saved the new contact in **ContactManager**.
12. Click the **Address Book** tab, and then search for the company you just created.
13. Click the company name in the search results to open its **Basics** card.
14. Look for the **Service States** list view above the **Notes**, and see that they are the states you added.
15. Click the **Edit in ContactManager** button, and, if necessary, log in as a user who can create new contacts.
For example, log in as user `aapplegate` with password `gw`.
ContactManager opens showing the company you created in **ClaimCenter**.

16. In ContactManager, verify that there is a **Service States** list above **Notes**, and that the states are the same as the ones you added in ClaimCenter.
17. Click **Edit** and delete one of the states.
18. Add a different state, and then click **Update**.
19. Click **Actions**→**New Company**→**Vendor** and choose each kind of company in turn to verify that there is a **Service States** list for each type of company.
20. Choose a type of company to create, then add and delete a service state. Add several service states, and then enter enough information to create the new Company.
21. Click **Update** to save the company.
22. Go back to ClaimCenter and click the **Address Book** tab, and then search for the company you created in ContactManager to verify that ClaimCenter can find the company.
23. In the claim in which you created the company with states, click **Parties Involved**.
24. On the **Contacts** screen, click the company you created and verify that it is now out of sync.
25. Click **Copy from Address Book** and verify that the company now has the set of states that you set up in ContactManager.

Changing the subtype of a Contact instance

You can change the subtype of a contact instance by using a command-prompt utility.

Important considerations before changing a contact subtype

This feature must be used with caution by experienced database and configuration professionals who are familiar with the Contact data model. It applies only to ContactManager and ClaimCenter. To use this feature, you must log out of the application and set the application server run level to **maintenance** during the contact subtype change.

The change of subtype directly affects the database. It causes ContactManager to be unable to synchronize the contact with ClaimCenter until you make the same subtype change in both ClaimCenter and ContactManager. Additionally, fields can be deleted and array fields can cause user interface exceptions.

For example:

- A subtype change can delete fields that do not exist in the new subtype. If feasible, determine which fields will change and where you want to move data in the new subtype. Then make any data updates you can in ContactManager before changing the subtype. You might also want to save the original contact data until you can review the contact after the subtype change.
- Related contacts and other arrays are not deleted when you change a subtype. However, they can cause problems in the user interface.
- In ContactManager, you can delete problematic data for the contact, such as related contacts that are not compatible with the new subtype, before making the subtype change.

See also

- “Contact subtype field differences and changing subtypes” on page 187

Overview

In ClaimCenter and ContactManager, you can change the subtype of a contact that was created with the wrong subtype without having to delete and re-create the contact. For example, you created a contact with subtype **Doctor** when you intended the contact to be subtype **Attorney**. If it is possible to delete the contact and re-create it with the correct subtype, do so. However, deleting and re-creating the contact might not be practical if the contact has history, such as being the payee on a check.

Changing subtype with a command-prompt utility

Changing the contact subtype is available through a command-prompt utility available in ContactManager and in ClaimCenter:

```

maintenance_tools
  -user user-name
  -password user-password
  -changesubtypetargettype subtype
  -changesubtypepublicid publicID

```

Note: Do not copy this command and paste it directly into the command line. You must first get rid of the linefeed characters that were inserted in this example for legibility.

You run this command from a command prompt open in the admin/bin folder of the installed product.

This command takes the following parameters:

user-name

User name of a user who has the role Contact Subtype Changer. This role includes the permissions required to run this command, `changecontactsubtype` and `soapadmin`. If you do not specify the `-user` parameter, the command defaults to Super User and requires that user's password.

user-password

Password for the user specified in the `-user` parameter, or for the Super User if there is no user specified.

subtype

New Contact or ABContact subtype for the contact, depending on whether you are running the command in ClaimCenter or ContactManager.

publicID

Value of the `PublicID` property of the contact. The value of `PublicID` is not necessarily the same for a contact stored in ContactManager and the equivalent contact instance or instances stored in ClaimCenter. Additionally, the value of `PublicID` is not necessarily the same as that of the `LinkID` in ContactManager or the `AddressBookUID` in ClaimCenter.

See also

- “Restrictions on changing the subtype of a contact instance” on page 186
- “ContactManager link IDs and comparison to other IDs” on page 260
- “Troubleshooting the change subtype command-prompt utility” on page 190

Restrictions on changing the subtype of a contact instance

There is information you must consider before changing the subtype of a contact instance, such as:

- Which applications store the contact
- Restrictions on client contact subtype
- Differences in fields supported by various contact subtypes

Contact instance location and changing subtype

The instance of a contact for which you want to change the subtype can be stored either in ClaimCenter or in ContactManager, or in both. In ClaimCenter, the contact would be on at least one claim's **Parties Involved**→**Contacts** page. If the ClaimCenter contact is linked to ContactManager, you must make the subtype change separately in each application.

There can be multiple instances of the contact in ClaimCenter, each stored as a claim contact with a claim. If contacts are not linked to ContactManager, you can change any number of them, based on the reason for the subtype change.

If the contacts are linked with ContactManager, you need to change all the linked contact instances in ClaimCenter and the single instance in ContactManager. One way to find the ClaimCenter instances is to first get the value of the `LinkID` of the contact in ContactManager. Then use that value in a query that uses the equivalent ClaimCenter contact property, `AddressBookUID`, to find all the contact instances in ClaimCenter. The value that you need from

each contact to make the subtype change is the `PublicID`. You then set the ClaimCenter run level to `maintenance` and run the command-prompt tool for each instance of the contact, specifying the value of its `PublicID`.

To change a contact subtype in ContactManager, set the run level to `maintenance` and then run the command-prompt tool once for the contact instance, using its `PublicID` to change its subtype. If the contact is linked to ClaimCenter, you set the run levels for both applications to `maintenance` and perform the subtype changes at the same time.

Subtype changes for client contacts

If the contact has the `Client` tag, there are restrictions on which subtypes you can change it to. Contacts with the `Client` tag cannot have their subtypes changed across the three main contact subtypes, `Person`, `Company`, and `Place`. This restriction is in place because neither PolicyCenter nor BillingCenter can handle such changes.

For example, if you try to make a subtype change in ContactManager for a client contact of type `ABDoctor` to the subtype `ABMedicalCareOrg`, the command fails with an error message. Because `ABDoctor` is an `ABPerson` subtype, the client contact's subtype cannot be changed to `ABMedicalCareOrg`, which is an `ABCompany` subtype.

If your subtype change stays within a single subtype hierarchy, such as `ABPerson` in ContactManager and `Person` in ClaimCenter, you are allowed to make the change. The same is true for the `ABCompany` and `Company` subtype hierarchies and the `ABPlace` and `Place` subtype hierarchies in ContactManager and ClaimCenter.

PolicyCenter and BillingCenter work only with `Company` and `Person` subtypes. Subtypes of `Person` all look the same to these applications, as do subtypes of `Company`. For example, a subtype change from a client contact of type `ABDoctor` to the subtype `ABAttorney` is permitted, because PolicyCenter and BillingCenter work with the contact as a `Person` subtype.

See also

- For information on contact subtypes in the base configuration, see “ContactManager and core application contact entity hierarchies” on page 147.

Contact subtype field differences and changing subtypes

Changing the subtype of a contact instance can cause some predictable changes in contact data. This topic describes some common changes. Before making the subtype change, you can save the current data for the contact, and then add data to the new subtype if that subtype supports the data. After the subtype change, you can see which fields were changed in the `Notes` field for the contact.

Name field changes between a company or place and a person

A `Person` subtype in the `en_US` locale can have first name, last name, prefix, and suffix fields, while a `Company` or `Place` subtype has only a name field. If you change the subtype from a person subtype to a company or place subtype, the person's first name, last name, prefix, and suffix are combined into a single name. They become the name of the company or place subtype. If you make the opposite subtype change, the name of the company or place subtype becomes the last name of the person subtype, with no first name.

Note: The set of name fields for a `Person` subtype can vary by locale. However, for those locales, the conversion works the same way between the multiple name fields of a `Person` subtype and the single name field of a `Company` subtype.

For example, you change the subtype of an instance of `Doctor` with prefix `Dr`, first name `Samantha`, and last name `Andrews` to `MedicalCareOrg`. The `Name` field of the `MedicalCareOrg` contact instance becomes `Dr Samantha Andrews`.

Subtype changes and incompatible data

If there are fields on one subtype that are not on the new one, the fields are dropped or converted to similar fields as part of the subtype change. However, array fields are not dropped, and if they are not compatible with the new subtype, that can cause problems in the user interface.

For example, a contact can have related contacts, such as a company that has employees, or a person who has an employer. However, in the base configuration, a company cannot have an employer, and a person cannot have

employees. If you change a person who has an employer to a company subtype, the employer field is preserved. However, the next time you edit the contact, you see an exception saying that a company cannot have an employer. If you go to the **Related Contacts** card and delete that relationship, you can then save the contact.

Additionally, if the person had the primary phone defined as a cell phone, that field is deleted during the transfer. In the base configuration, the primary contact number for a company contact cannot be a cell phone.

You can compare the fields for the subtypes in the Data Dictionary. If there are fields that will be dropped, you can record the data. After you change the subtype, you can add data that is appropriate for the new subtype. You can also consider deleting fields that will cause problems before making the contact subtype change.

Changing the subtype of a contact instance

Be sure to read the important note at the beginning of the main topic and the restrictions on changing subtypes before starting this multi-step, multi-topic process. In particular, if the contact has the Client tag, you cannot change the subtype between Person, Company, or Place subtypes. For example, you cannot change the subtype of a Client contact from Doctor, a Person subtype, to MedicalCareOrg, a Company subtype. The tool will prevent you from making this change.

- For an Important note, see “Changing the subtype of a Contact instance” on page 185.
- For information you need to know before you start, see “Restrictions on changing the subtype of a contact instance” on page 186

If there is data in the contact that will be lost or will not be compatible with the new subtype, if possible, fix the data before you change the subtype. If the contact is stored in ClaimCenter and is linked to ContactManager, first make the data changes to the contact in ContactManager. The changes are then propagated to ClaimCenter, which updates all linked instances of the contact. After you complete your data changes, you can proceed with the contact subtype change.

Note: It is not always possible to make all the data changes before the subtype change, so you might have to make some data changes afterwards.

You can change the subtype of a contact stored only in ClaimCenter, stored only in ContactManager, or stored in both and linked together. The server runlevel must be set to *maintenance* to make the change. If the contact is stored in both ClaimCenter and ContactManager, set both servers’ runlevels to *maintenance*. Leave them at *maintenance* runlevel until you complete the subtype changes and any subsequent edit verifications in both applications.

Change the subtype of an ABContact instance in ContactManager

Before you begin

There are limitations on changing the subtype of a contact instance that you must be aware of before starting this process. See “Changing the subtype of a contact instance” on page 188.

Procedure

1. Run a query on the ContactManager database to get the PublicID and LinkID of the contact instance.
2. Ensure that all users are logged out.

Note: This step is an important one. ContactManager must not have its user interface open when you apply the subtype change.

3. Set the ContactManager server’s runlevel to *maintenance*.

For example, at a command prompt open in ContactManager/admin/bin, enter:

```
system_tools -password adminuser-password -maintenance
```

4. Run the command-prompt tool to change the subtype of the contact instance.

For example, the user with login alinu has the role Contact Subtype Changer and password x2wTz@71P. For user alinu to change the contact with PublicID ab:123 to the subtype ABDirector, at a command line open in ContactManager/admin/bin, the user would enter the following command, all on one line:

```
maintenance_tools -user alinu
                  -password x2wTz@71P
```

```
-changesubtypetargettype ABDoctor  
-changesubtypepublicid ab:123
```

5. Note the messages in the console.

If your command is successful, you see messages similar to the following:

```
Changing contact with publicID: ab:123 to subtype: ABDoctor  
done
```

6. Log in to ContactManager and ensure that you can open the contact and save changes. If necessary, update data as needed.
7. Take ContactManager out of maintenance runlevel.

For example, at a command prompt open in ContactManager/admin/bin, enter:

```
system_tools -password adminuser-password -maintenance
```

Next steps

“Change the subtype of a Contact instance in ClaimCenter” on page 189

Change the subtype of a Contact instance in ClaimCenter

Before you begin

Complete “Change the subtype of an ABContact instance in ContactManager” on page 188.

Procedure

1. Make a query for contact data. You might proceed differently depending on whether the contact is linked to ContactManager or is just local to ClaimCenter:
 - If the contact is linked to ContactManager, run a query to find all instances of the contact with the AddressBookUID value equal to the LinkID value you got from ContactManager. Save the PublicID of each contact instance that the query finds. Additionally, note each claim for which the contact is a claim contact.
 - If the contact is local only to ClaimCenter, run a query in ClaimCenter to find the contact and save its PublicID. Note the claim for which the contact is a claim contact. You can run multiple queries if there are duplicate contact instances on other claims that you want to change.

2. Set the ClaimCenter server's runlevel to maintenance.

For example, at a command prompt open in ClaimCenter/admin/bin, enter:

```
system_tools -password adminuser-password -maintenance
```

3. Run the command-prompt tool for each contact instance your query found.

For example, the user with login alinu has the role Contact Subtype Changer and password x2wTz@71P. For user alinu to change the contact with PublicID ab:123 to the subtype Doctor, at a command line open in ContactManager/admin/bin, the user would enter the following command, all on one line:

```
maintenance_tools -user alinu -password x2wTz@71P  
-changesubtypetargettype Doctor  
-changesubtypepublicid ab:123
```

4. Note the messages in the console.

For each command that completes successfully, you see messages similar to the following:

```
Changing contact with publicID: ab:123 to subtype: Doctor  
done
```

5. Log in to ClaimCenter and ensure that you can open the contact in each claim and save changes. If necessary, update data as needed.
6. Take ClaimCenter out of maintenance runlevel.

For example, at a command line open in ClaimCenter/admin/bin, enter:

```
system_tools -password adminuser-password -multiuser
```

Troubleshooting the change subtype command-prompt utility

The command-prompt tool that changes the subtype of a contact instance is described at “Changing subtype with a command-prompt utility” on page 185. When you run this command, the following errors can be reported in the console.

Error message	Cause of the error
Failed to change contact to subtype: Bad username or password	<p>Any of the following entries could have caused this problem:</p> <ul style="list-style-type: none"> You entered an invalid user name. You entered the wrong password. You entered the password without a user specified and you are not using the password of the Super User. <p>Unless you are user Super User, the default user for this command, you must use the <code>-user</code> parameter and specify a user that has the permissions <code>changecontactsubtype</code> and <code>soapadmin</code>. For example, the role Contact Subtype Changer has these two permissions.</p>
Failed to change contact to subtype: Incorrect runlevel for subtype change. Required: MAINTENANCE, Actual: MULTIUSER	<p>You have not set the application runlevel to maintenance prior to running the command. Use the following command:</p> <pre>system_tools -password adminuser-password -maintenance</pre>
Failed to change contact to subtype: Contact with PublicID <i>public-id-value</i> not found	You entered an incorrect value for the PublicID of the contact.
Failed to change contact to subtype: <i>entity-name</i> is not a valid entity type	The entity name you used is not valid. For example, you used ABDoctor in ClaimCenter, or Doctor in ContactManager.
Failed to change contact to subtype: <i>entity-name</i> is not a valid subtype of Contact	The entity name you used for the new subtype is not a subtype of ABContact or Contact. For example, you tried to change the contact subtype from ABPerson to ABContact.
Failed to change contact to subtype: Cannot change Person to Company/Place or Company to Person/Place for contacts with Client tags	The contact has a Client tag, which makes the contact usable by PolicyCenter and BillingCenter. Those applications do not support changing between a Person contact subtype and a Company or Place subtype. See “Subtype changes for client contacts” on page 187.
You see the list of command-prompt options for <code>maintenance_tools</code> , but no error message	Either you misspelled one of the command-prompt options or you did not enter one. Changing a contact subtype requires that you enter both options, <code>-changesubtypetargettype</code> and <code>-changesubtypepublicid</code> . See “Changing the subtype of a Contact instance” on page 185.

Contact tags

Contact tags enable you to classify contacts without having to add new subtypes. Every contact stored by ContactManager must have at least one tag. The three contact tag types provided in the base configuration are:

Client

A policy or account contact in PolicyCenter. For example, the holder of an account, the primary named insured on a policy, or a driver of a vehicle insured by a policy.

Vendor

ClaimCenter vendor providing services that help resolve claim losses. For example, a body shop, assessor, attorney, or physical therapy clinic.

Claim Party

A party to a claim in ClaimCenter, such as the insured or a claimant.

These tags are used to ensure that a contact is either a vendor, and therefore of interest only to ClaimCenter, or a client, of interest to PolicyCenter. A client can also be of interest to ClaimCenter and BillingCenter. Additionally, the Claim Party tag indicates that a contact has been added as a party to a claim. You can add additional tags and change how the applications use them.

IMPORTANT Do not remove or edit the three contact tag types provided in the base configuration in the ContactTagType typelist. Any changes to these three tag types can cause your contact system integration to stop working. You can add new contact tag types to the base typelist in Guidewire Studio™. Have a good reason for doing so. If the new additions are not related to identifying contacts for searches, consider adding a separate array of identifiers rather than extending this typelist.

Applications and contact tags

The base configurations of PolicyCenter and BillingCenter save and retrieve only ContactManager contacts that have the Client tag. For example, a PolicyCenter user adds a client contact to a policy. PolicyCenter automatically sets the tag to Client and saves the contact in ContactManager. Later a claims adjuster adds this client contact to a claim as a party to the claim and saves the contact. ClaimCenter applies the Claim Party tag to the contact before sending it to ContactManager. The contact now has both the Client tag and the Claim Party tag.

When you add a new contact or edit a contact in ContactManager, you see a drop-down list that enables you to choose one or more contact tags. Additionally, you can see the contact tag on the detail screen for every contact.

When you work with contacts in the base configurations of ClaimCenter, PolicyCenter, and BillingCenter, you do not see any tag information in the user interface. The applications work with contact tags in the background and add the appropriate tags before sending a contact to ContactManager. After the tags are added to the contact in

ContactManager, those tags are synchronized as part of the contact data copied over from ContactManager to the local contact record.

Note: A contact saved only locally in a core application does not have to have a tag. However, in their base configurations, the core applications always set tags for contacts when they are created and when they are saved to the database. Core applications set tags for contacts that they store locally even if the contacts are not stored in ContactManager.

Add a new contact tag

About this task

You can add a new contact tag type.

Procedure

1. Navigate in the Guidewire Studio™ **Project** window to **configuration→config→Metadata→Typelist**.
2. Right-click `ContactTagType.tti`.
3. Choose **New→Typelist Extension**.
If you get a message saying that you cannot create a typelist from `ContactTagType.tti`, then `ContactTagType.ttx` already exists. You can open it in **configuration→config→Extensions→Typelist**.
4. If you do not get an error message, click **OK**.
Guidewire Studio™ creates the file `ContactTagType.ttx` in **configuration→config→Extensions→Typelist**.

ClaimCenter contact tag generation

If a new contact created in ClaimCenter is a vendor subtype, ClaimCenter adds a Vendor tag when it saves the contact to ContactManager. If you add a contact to a claim, ClaimCenter ensures that it has the Claim Party tag before saving the contact in ContactManager.

The base configuration of ClaimCenter sets tags in the class `gw.api.contact.CCContactMinimalTagsImpl`. If you add new tags, you must also set them in this class.

PolicyCenter contact tag generation

PolicyCenter ensures that any contacts it sends to ContactManager have Client tags. PolicyCenter cannot find a contact in ContactManager unless the contact has a Client tag.

The base configuration of PolicyCenter sets tags in the class `gw.api.contact.PCContactLifecycle`. If you add new tags, you must also set them in this class.

BillingCenter contact tag generation

BillingCenter ensures that any contacts it sends to ContactManager have Client tags. BillingCenter cannot find a contact in ContactManager unless the contact has a Client tag.

The base configuration of BillingCenter sets tags in the Contact Preupdate rule **Add default tags**. If you add new tags and want to set them for contacts that BillingCenter sends to ContactManager, you must set them in this rule.

Contact tag-based security

There are permissions associated with tags and there are permission check expressions used to control access to screens.

Contact tag permissions

The base application permissions, listed in the table that follows, are special permissions that enable a user to create, edit, delete, and view contacts that have any tag. You can add permissions for creating, editing, deleting, and viewing specific tags, just as you can for contact subtypes.

The tag permissions provided in the base configurations are:

Code	Permission Description
anytagcreate	Create a new contact regardless of which contact tag it requires.
anytagedit	Edit a contact that has any contact tag.
anytagdelete	Delete a contact that has any contact tag.
anytagview	See a contact that has any contact tag.

See also

- For more information on contact tag permissions, see:
 - “ContactManager contact and tag permission check expressions” on page 125
 - “ClaimCenter contact subtype and tag permissions” on page 134
 - “BillingCenter contact-related permissions” on page 132
- You can associate permissions with tags in Guidewire Studio™. For information on adding new tag permissions, see:
 - “Creating client tag permissions in ContactManager” on page 129
 - “Create claim party and vendor tag permissions in ClaimCenter” on page 140

Contact tag permission check expressions

Guidewire applications use a set of permission check expressions to control access to screens and widgets related to contacts. In ContactManager, to perform an action, users need permission for both the contact's subtype and the contact's tags.

For example, to edit a *CompanyVendor* contact, the user needs permission to edit contacts of that subtype. If the contact has the *Vendor* tag, the user also needs permission to edit contacts with the *Vendor* tag. If you have not extended the tag permissions, the base configuration *anytagedit* permission gives that user permission to edit vendor contacts.

See also

- “ContactManager contact and tag permission check expressions” on page 125
- “ClaimCenter contact and tag permission check expressions” on page 135
- “BillingCenter contact and tag permission check expressions” on page 133

Linking and synchronizing contacts

You can set up links for contacts and synchronize contact information between Guidewire core applications and the Guidewire ContactManager™ contact management system. You must be managing contacts in an environment where ContactManager is integrated with your Guidewire core applications.

Note: This topic covers synchronizing contact data between a Guidewire core application and ContactManager, which is more complex and configurable in ClaimCenter than it is in PolicyCenter or BillingCenter. From the point of view of the core application, this type of synchronization is *external contact synchronization*. PolicyCenter also defines behavior for synchronizing contact data internally between accounts and policies. That type of synchronization is distinct from external contact synchronization.

See also

- “Locally and centrally managed contacts” on page 52

Linking a contact

Before contact information can be synchronized between ContactManager and a Guidewire core application, it must be linked. Linking ensures that a contact stored locally in the core application is connected to a contact stored in ContactManager. ContactManager uniquely identifies an `ABContact` instance by its `LinkID`. The core applications use `AddressBookUID` for the same purpose. A contact is linked when the core application has verified the contact’s `AddressBookUID` with ContactManager.

Creating and linking a contact

The initial linking of a core application contact and a ContactManager contact typically happens when the core application creates the contact and sends it to ContactManager. In the base configuration, PolicyCenter specifies the unique ID for a new contact, while ClaimCenter and BillingCenter let ContactManager specify the unique ID for a new contact. In all three core applications, `AddressBookUID` holds the unique ID for the contact.

- In the base configuration, ClaimCenter and BillingCenter send the `createContact` request to ContactManager, and ContactManager creates a unique `LinkID` for the contact. ContactManager then returns the new contact data, including the `LinkID`, to the calling application. ContactManager also broadcasts the new contact data to the other applications.
- In the base configuration, PolicyCenter generates a unique ID for the new contact and sends the `createContact` request to ContactManager. ContactManager populates `LinkID` for the new contact with the unique ID specified by PolicyCenter.

See also

- “Linking in ClaimCenter” on page 197
- “Linking in BillingCenter” on page 199
- “PolicyCenter contact creation with external unique IDs” on page 196
- “Linking in PolicyCenter” on page 198

PolicyCenter contact creation with external unique IDs

PolicyCenter must be able to create a new contact and send it both to BillingCenter and ContactManager. PolicyCenter must be able to uniquely identify a contact it sends to BillingCenter, and BillingCenter must be able to link to the contact created in ContactManager. Therefore, in the base configuration, PolicyCenter specifies a unique identifier for any contact it creates and sends to ContactManager.

If PolicyCenter needs to notify BillingCenter that the new contact has been created, PolicyCenter also sends a message to BillingCenter indicating the `AddressBookUID` of the new contact. The value of the `AddressBookUID` is the unique identifier that PolicyCenter created for the contact.

These messages are asynchronous and can arrive at BillingCenter and ContactManager at different times and in no particular order.

When ContactManager receives a contact create request for which a unique ID is specified, it uses that ID for the `LinkID` value of the new contact. ContactManager then sends create messages for update to BillingCenter and ClaimCenter if they are registered.

Depending on when BillingCenter receives the messages, it proceeds down different paths to link the contact:

- BillingCenter receives the message from PolicyCenter before ContactManager creates the new contact.
 1. BillingCenter sends a request to ContactManager for contact data, specifying the `AddressBookUID`.
 2. ContactManager returns `null`, indicating that the contact does not exist.
 3. BillingCenter creates a local copy of the contact with the `AddressBookUID` it received from PolicyCenter.
 4. ContactManager later notifies BillingCenter that the contact was created.
 5. BillingCenter verifies that the `LinkID` sent by ContactManager matches the `AddressBookUID` of the local contact that it created.
 6. BillingCenter updates its local copy of the contact with the data from ContactManager and links the contact. See “Linking in BillingCenter” on page 199.
- BillingCenter receives the message from ContactManager about the new contact before it receives the message from PolicyCenter.
 1. BillingCenter ignores the message from ContactManager, because it does not have a matching contact.
 2. BillingCenter receives a message from PolicyCenter saying that there is a new contact with a specified `AddressBookUID`.
 3. BillingCenter sends a request to ContactManager for contact data, specifying the `AddressBookUID`.
 4. ContactManager sends the data for the contact whose `LinkID` matches the `AddressBookUID` sent by BillingCenter.
 5. BillingCenter updates its local copy of the contact with the data from ContactManager and links the contact. See “Linking in BillingCenter” on page 199.

See also

“PolicyCenter support for creating external unique IDs” on page 197

ContactManager support for handling external unique IDs

ContactManager provides support for handling external unique IDs as follows:

- The `ContactMapper` class in ContactManager has field mapping definitions for the `External_UniqueID` fields in each entity that implements `ABLinkable`. These fields are listed in the topic “Mapping externally specified unique IDs of a ContactManager contact” on page 277.
- If a `createContact` request comes in that specifies an `External_UniqueID` field for the new contact, ContactManager uses the value of that field to populate the `LinkID` for the new contact. If that field is not in the

data for the new contact or the field exists but its value is null, ContactManager creates its own LinkID for the new contact. See the description of the createContact method in “ABContactAPI methods” on page 264.

- ContactManager provides a CreatedApproved rule in the EventFired ruleset for each core application. The rule is fired by the event ABContactCreatedApproved, which is added to an ABContact when it is created with an approved status by a call to ABContactAPI. The rule causes ContactManager to broadcast a create message as an update to each configured core application. This message enables any application that has been notified of a new contact that has an external ID provided to ContactManager to receive an update message when the create happens.

For example, PolicyCenter sends a new contact to BillingCenter with the AddressBookUID specified. When ContactManager creates the new contact, BillingCenter gets the Create for Update message from ContactManager. BillingCenter can then compare the LinkID in the message with the AddressBookUID of the contact that PolicyCenter sent to BillingCenter and update the contact with the data from ContactManager.

PolicyCenter support for creating external unique IDs

PolicyCenter provides support for creating external unique IDs with the following:

- The Contact entity in PolicyCenter is extended with the field ExternalID, which stores the value of the unique ID that PolicyCenter generates.
- The ContactMapper class in PolicyCenter has field mapping definitions for various EXTERNAL_UNIQUE_ID fields. Those definitions are described in “PolicyCenter mapping of externally specified unique IDs” on page 281.
- PolicyCenter has code that generates a new unique ID and assigns it to the ExternalID field of a new contact. PolicyCenter then calls the ABContactAPI.createContact method, passing it the data for that new contact. See the class `gw.api.contact.PCContactLifecycle`.
- PolicyCenter has an EventFired rule that responds to a ContactAdded event and sends a newly created contact to BillingCenter.

Linking in ClaimCenter

ClaimCenter links a vendor contact automatically with ContactManager if the contact is created by a user that has the permissions `abcreate` or `abedit`. If the user does not have these permissions, in the base configuration, new vendor contacts are created, flagged as pending in ContactManager, and linked. These pending creates must be approved by a ContactManager user. After approval, ContactManager removes the pending flag.

This ClaimCenter behavior with vendor contacts is defined in the Gosu class `gw.plugin.contact.ab1000.ContactSystemApprovalUtil`. You can edit this class and change how ClaimCenter determines the following:

- If a contact created in ClaimCenter will be created in ContactManager
- If a contact created or updated in ClaimCenter and sent ContactManager requires approval

In ClaimCenter, you can also manually link a local contact to ContactManager. For example, on the **Parties Involved** screen of a claim, you can click the name of a local contact and, below it in the **Basics** tab, click **Link**. After you click **Link**, ClaimCenter calls the ContactManager `findDuplicates` web service to see if there is a matching contact.

Note: In the base configuration, this link functionality uses the `WIDE_MAP` variable to try to find a matching contact. See “IFindDuplicatesPlugin plugin interface” on page 289.

Depending on the results of the `findDuplicates` call, ContactManager and ClaimCenter behave as follows:

- If ContactManager finds an exact match, ClaimCenter links the local contact to the ContactManager contact by populating the local contact's `AddressBookUID` with the `LinkID` of the ContactManager contact.
- If ContactManager finds potential matches, ClaimCenter enables you to select one of the potential matches.
- If ContactManager does not find any matches, its behavior depends on the permissions of the ClaimCenter user and the type of contact:
 - If the ClaimCenter user has the `abcreate` permission, ClaimCenter sends a request to ContactManager to create a new contact.
 - If the ClaimCenter user does not have the `abcreate` permission and the contact is a vendor contact, ClaimCenter sends the new vendor to ContactManager with pending status. This new vendor must be approved by a user logged in to ContactManager.
 - If the contact is not a vendor contact, such as `Person`, ClaimCenter sends a request to ContactManager to create the new contact. The ClaimCenter user does not have to have `abcreate` permission.

After a successful link, the contact becomes subject to synchronizing rules, and the **Link** button turns into an **Unlink** button. If you click **Unlink**, the contact is no longer connected to ContactManager. It becomes a local-only contact.

See also

- “Synchronizing ClaimCenter and ContactManager contacts” on page 203
- “Find duplicates behavior” on page 199

Linking in PolicyCenter

In the base configuration, PolicyCenter links contacts with an integrated ContactManager under the following conditions:

- PolicyCenter always links contacts that are on accounts with at least one bound policy. At the time the user adds the contact, PolicyCenter checks to see if there are possible duplicate contacts.
- If the contact initially comes from ContactManager, PolicyCenter links the local contact when it creates the local contact. For example:
 - If you search for contacts and add one that is in ContactManager, that contact becomes linked regardless of the state of the policies on the account.
 - If you click to add a new contact, ContactManager can return a list of exact and potential duplicates. If you choose one of the duplicates as the contact that you want to use, that contact is linked.

PolicyCenter does not send new client data to ContactManager while the client is still a prospect—when there is no bound policy on the account. PolicyCenter does store prospect information locally. If a prospect becomes a customer—at least one policy is bound for the account—PolicyCenter links the contact to the contact management system. You can configure this behavior differently.

The `Contact` entity and its subentities have an `AutoSync` property that controls synchronization with ContactManager. Setting this property to `Allow` enables the contact to be synchronized. After a contact is linked, PolicyCenter ensures that the `AutoSync` property is set to `Allow`.

See also

- “Find duplicates behavior” on page 199
- “Configuring PolicyCenter external contact synchronization” on page 201

Linking in BillingCenter

In the base configuration, BillingCenter links contacts with an integrated ContactManager under the following conditions:

- BillingCenter links contacts that are on accounts and policy periods. At the time the user adds the contact, BillingCenter can check to see if there are possible duplicate contacts.
 - If you click **Add Existing Contact** for an account or policy period and search for a contact, BillingCenter shows a list of matching internal and external contacts.
 - If you add an external contact, one that is stored only in ContactManager, BillingCenter creates a local instance of the contact and links it.
 - If you add a contact that is not external, the contact has a local instance that BillingCenter might already have linked. If the contact instance is not linked, BillingCenter saves it as a new contact in ContactManager and links it.
 - If you click to add a new contact, BillingCenter checks to see if it exists in ContactManager. This check can return a list of exact and potential duplicates. If you choose one of the duplicates as the contact that you want to use, that contact is linked. If you create the contact without choosing from the list, BillingCenter sends it to ContactManager as a new contact, and at that point it gets linked.
- For producer contacts, BillingCenter creates a local instance of the contact. It does not link producer contacts with ContactManager.

BillingCenter find duplicates behavior is defined in the method

`gw.plugin.contact.ab1000.ABContactSystemPlugin.findDuplicates`. For example, this method specifies that the only tag to be used to find duplicates is the client tag. If you want to use other tags for finding duplicates, you can modify this method to do so.

The `Contact` entity and its subentities have an `AutoSync` property that affects synchronization with ContactManager. Setting this property to `Allow` for a contact of an account or policy period enables the contact to be synchronized. BillingCenter sets the `AutoSync` property to `Allow` for all new contacts.

See also

- “Find duplicates behavior” on page 199
- “Configuring BillingCenter external contact synchronization” on page 202

Find duplicates behavior

In their base configurations, the Guidewire core applications call the ContactManager web service `ABContactAPI.findDuplicates` to see if there are duplicates for a contact. The match types that ContactManager can return are *exact match* and *potential match*. For example:

- ClaimCenter calls this API when setting up a link for a contact. If there is an exact match, ContactManager establishes the link automatically. If the match is potential, ContactManager sends the potential matches to ClaimCenter, which displays them and enables you to choose one. Linking proceeds as described in “Linking in ClaimCenter” on page 197.
- Both PolicyCenter and ClaimCenter call this API when you click the **Check for Duplicates** button while adding a new contact. They also call this API when you save a new contact and you have not previously clicked **Check for Duplicates**.
 - If you click **Check for Duplicates**, ContactManager checks for exact and potential matches and sends the results to the core application, PolicyCenter or ClaimCenter. The core application shows the results and enables you to choose one.
 - If you click to save a new contact and have not clicked **Check for Duplicates**, the core application and ContactManager participate in the find duplicates exchange as described previously.
 - If you click to save a new contact and have already clicked **Check for Duplicates**, there is no additional find duplicates check. However, if there is an exact match that you have not chosen, the core application shows a warning that an exact match was available.

You must confirm one of the following:

- If you want to create a new contact, confirm that you do not want to choose the exact match.
- If you do not want to create a new contact, click to cancel the operation.
- BillingCenter calls this API when you add a new contact to an account or policy period, as follows:
 - The first time you click to save a new contact, BillingCenter and ContactManager participate in a find duplicates exchange. ContactManager checks for exact and potential matches and sends the results to BillingCenter. BillingCenter shows the results and enables you to choose one.
 - If you choose a contact from a list of exact or potential matches, BillingCenter links that contact and sets it up to be synchronized.
 - You might save a new contact if there were no matches listed or if you do not see a good match on the list of contacts. In this case, there is no additional find duplicates check. However, if there is an exact or potential match on the list that you have not chosen, BillingCenter shows a warning that a matching contact was available.

You must click one of the following:

- **Update** to continue and create the new contact
- **Check for Duplicates** to choose from the list.
- In ContactManager, to configure this matching behavior you can edit the `FindDuplicatesPlugin` plugin, which implements the interface `IFindDuplicatesPlugin`. Additionally, there is a set of Duplicate Finder classes that work with the plugin. You can edit these classes to add new subtypes and fields to the find duplicates process.

See also

- “IFindDuplicatesPlugin plugin interface” on page 289
- “ABContactAPI web service” on page 264

Synchronizing with ContactManager

After a contact has been linked with ContactManager, the core applications and ContactManager work together to keep the contact data synchronized. This topic describes how contacts are synchronized, how the applications communicate changes to contact data, and how they ensure that this data is kept current across your Guidewire applications.

ContactManager plugins for broadcasting of contact changes

ContactManager uses three plugins and an API to broadcast contact updates and changes to the Guidewire core applications. The registries for the plugins are `ClaimSystemPlugin.gwp`, `PolicySystemPlugin.gwp`, and `BillingSystemPlugin.gwp`. The base ContactManager application provides plugin implementations that ContactManager can use to communicate with ClaimCenter, PolicyCenter, and BillingCenter. The plugin implementations are:

- `gw.plugin.claim.cc1000.CCClaimSystemPlugin`
- `gw.plugin.policy.pc1000.PCPolicySystemPlugin`
- `gw.plugin.billing.bc1000.BCBillingSystemPlugin`

To enable ContactManager to communicate with the Guidewire core applications you have installed, you configure the application plugin registry corresponding to each installed application by specifying the corresponding plugin implementation. Then ContactManager can then send updates by calling each application’s `ContactAPI` web service, which implements the interface `ABClientAPI`. At that point, the application handles the update in its `ContactAPI` implementation of `ABClientAPI`.

See also

- “Integrating ContactManager with ClaimCenter in QuickStart” on page 59
- “Integrating ContactManager with PolicyCenter in QuickStart” on page 66

Synchronizing PolicyCenter and ContactManager contacts

PolicyCenter synchronizes contact data with ContactManager by using *external contact synchronization*, which is distinct from PolicyCenter internal synchronization. PolicyCenter separately defines behavior for synchronizing contact data internally between accounts and policies.

PolicyCenter performs external contact synchronization with ContactManager for contacts that are shared across accounts, not for contacts at the policy level. PolicyCenter synchronizes contact data between policies and accounts internally.

In the PolicyCenter integration with ContactManager, linked client data is synchronized with ContactManager at all times. For PolicyCenter, ContactManager is the system of record for linked client data.

PolicyCenter uses the class `gw.plugin.contact.ab1000.ABContactSystemPlugin` to create, retrieve, update, delete, and find duplicates of contacts in ContactManager. This class implements `gw.plugin.contact.ContactSystemPlugin` and calls the methods of the ContactManager web service `ABContactAPI`.

To enable ContactManager to send contact updates to PolicyCenter, PolicyCenter implements the ContactManager interface `ABClientAPI` in the class `gw.webservice.pc.pc1000.contact.ContactAPI`. ContactManager requires that all Guidewire core applications implement the `ABClientAPI` interface. Implementing this interface ensures that ContactManager can broadcast contact changes to the core applications by using standard web services, and that the core applications have logic to handle these updates.

Exceptions to PolicyCenter synchronization updates

There are some cases in which PolicyCenter cannot perform a contact change or update:

- ContactManager notifies PolicyCenter that a contact has been deleted, but the contact is being used on at least one account in PolicyCenter.

This case is not likely to happen, because ContactManager checks to see if a contact can be deleted before it sends the request to delete the contact to PolicyCenter. However, if this case does happen, PolicyCenter unlinks the contact, does not delete the local instance, and creates an activity warning that an external system tried to delete the contact.

- ContactManager notifies PolicyCenter that an address has been deleted, but the address is being used as a policy address in PolicyCenter.

In this case, PolicyCenter unlinks the local instance of the address and does not delete it.

- PolicyCenter sends changes to a contact's data to ContactManager, but the contact's data was changed previously by another application or directly in ContactManager. The original data for the contact in PolicyCenter, prior to the change, must match the data currently in ContactManager for the change to be accepted. Since the original data that PolicyCenter sent does not match, ContactManager sends an exception to PolicyCenter and does not update the contact's data.

In this case, PolicyCenter creates an activity for a user to reapply the updates to the contact's data. The activity contains information about the contact and the data that could not be applied in ContactManager.

Configuring PolicyCenter external contact synchronization

The PolicyCenter `Contact` entity has a typekey attribute called `AutoSync` that controls whether contacts are automatically synchronized with the external contact management system. Any PolicyCenter contact that is linked to ContactManager has the same value in its `AddressBookUID` property as the `LinkID` of the corresponding contact in ContactManager. Guidewire recommends that all contacts that are in both PolicyCenter and ContactManager be linked and have an `AutoSync` setting of `Allow`.

Note: In the default integration, PolicyCenter does not send a new contact to ContactManager until there is at least one bound policy on an account that the contact is associated with. When a contact is not linked to ContactManager, it has an `AutoSync` setting of `null`.

For any `Contact` instance, in addition to `null`, the `AutoSync` value can be one of three codes:

Allow	Allow the contact to be synchronized automatically. In a rule, this value would be <code>TC_ALLOW</code> .
-------	--

Disallow	Do not allow the contact to be synchronized. In a rule, this value would be TC_DISALLOW. In its base configuration, PolicyCenter never sets a contact to Disallow.
Suspended	The contact was synchronized in the past, but synchronizing is not currently allowed. In a rule, this value would be TC_SUSPENDED. For example, this value could be set for all contacts during a ContactManager outage.

PolicyCenter uses the Java method `AccountContactImpl.markContactForAutoSync` internally to set the `AutoSync` value on a contact. This method is scriptable. The method sets `AutoSync` to `Allow` only if it is not already set to `Disallow`. `Disallow` is treated as a terminal state.

Synchronizing PolicyCenter contact fields

When PolicyCenter synchronizes contact data with ContactManager, it updates the fields in the PolicyCenter contact with the centralized ContactManager data. The system overwrites all the fields defined in the `ContactMapper` class. This class also defines the fields that PolicyCenter sends to ContactManager. Use this class to control how contact data is synchronized between PolicyCenter and ContactManager.

See also

- “Working with contact mapping files” on page 150

Synchronizing BillingCenter and ContactManager contacts

BillingCenter performs contact synchronization with ContactManager for contacts of accounts and policy periods, but not for contacts of producers. For BillingCenter, ContactManager is the system of record for linked client data. BillingCenter uses the class `gw.plugin.contact.ab1000.ABContactSystemPlugin` to create, retrieve, update, delete, search for, and find duplicates of contacts in ContactManager. This class implements `ContactSystemPlugin` and calls the methods of the ContactManager web service `ABContactAPI`.

To enable ContactManager to send contact updates to BillingCenter, BillingCenter implements the ContactManager interface `ABClientAPI` in the class `gw.webservice.bc.bc1000.contact.ContactAPI`. ContactManager requires that all Guidewire core applications implement the `ABClientAPI` interface. Implementing this interface ensures that ContactManager can broadcast contact changes to the core applications by using standard web services, and that the core applications have logic to handle these updates.

Exceptions to BillingCenter synchronization updates

There are some cases in which BillingCenter cannot perform a contact change or update:

- ContactManager notifies BillingCenter that a contact has been deleted, but the contact is being used on at least one account in BillingCenter.
This case is not likely to happen, because ContactManager checks to see if a contact can be deleted before it sends the request to delete the contact to BillingCenter. However, if this case does happen, BillingCenter unlinks the contact and does not delete the local instance.
- BillingCenter sends changes to a contact’s data to ContactManager, but the contact’s data was changed previously by another application or directly in ContactManager. The original data for the contact in BillingCenter, prior to the change, must match the data currently in ContactManager for the change to be accepted. Since the original data that BillingCenter sent does not match, ContactManager sends an exception to BillingCenter and does not update the contact’s data.

In this case, BillingCenter creates an activity for a user to reapply the updates to the contact's data. The activity contains information about the contact and the data that could not be applied in ContactManager.

Configuring BillingCenter external contact synchronization

The BillingCenter `Contact` entity has a typekey attribute called `AutoSync` that is important in determining if contacts are automatically synchronized with the external contact management system. Any BillingCenter contact that is linked to ContactManager has the same value in its `AddressBookUID` property as the `LinkID` of the corresponding contact in ContactManager. BillingCenter sets `AutoSync` for all new contacts to `Allow`.

Note: In the default integration, BillingCenter does not synchronize contacts of producers. BillingCenter adds a property to Contact entities, `ShouldSendToContactSystem`, that determines if the contact is synchronized. This Boolean property is set in a `get` method in `gw.api.address.ContactEnhancement`, described later in this topic.

For any Contact instance, the `AutoSync` value can be one of three codes, in addition to `null`:

Allow	Allow the contact to be synchronized automatically. In a rule, this value would be <code>TC_ALLOW</code> .
Disallow	Do not allow the contact to be synchronized. In a rule, this value would be <code>TC_DISALLOW</code> . In its base configuration, BillingCenter never sets a contact to <code>Disallow</code> .
Suspended	The contact was synchronized in the past, but synchronizing is not currently allowed. In a rule, this value would be <code>TC_SUSPENDED</code> . For example, this value could be set for all contacts during a ContactManager outage.

The previous code descriptions provide values you can set for `AutoSync` in a rule. To see a rule that sets `AutoSync`, open Guidewire Studio™ for BillingCenter. Then, in the **Project** window, navigate to **configuration→config→Rule Sets→Preupdate→ContactPreupdate→All Contacts sync by default**.

BillingCenter does not link or synchronize contacts of producers. To control this aspect of synchronization, BillingCenter adds a property to Contact entities, `ShouldSendToContactSystem`, that determines if the contact is synchronized. This Boolean property is set in a `get` method in `gw.api.address.ContactEnhancement`. The code for the method is:

```
property get ShouldSendToContactSystem() : boolean {
    return this.AutoSync == AutoSync.TC_ALLOW
        and not this.ID.Temporary
        and (isOnAccount() or isOnPolicyPeriod())
}
```

For a contact to be sent to ContactManager, it must have an `AutoSync` value of `AutoSync.TC_ALLOW` and it must be connected to either an account or a policy period.

Synchronizing BillingCenter contact fields

When BillingCenter synchronizes contact data with ContactManager, it updates the fields in the BillingCenter contact with the centralized ContactManager data. The system overwrites all the fields defined in the `ContactMapper` class. This class also defines the fields that BillingCenter sends to ContactManager. Use this class to control specifics of synchronizing contact data between BillingCenter and ContactManager.

See also

- “Working with contact mapping files” on page 150

Synchronizing ClaimCenter and ContactManager contacts

To enable ContactManager to send contact updates to ClaimCenter, ClaimCenter implements the ContactManager interface `ABClientAPI` in the class `gw.webservice.cc.cc1000.contact.ContactAPI`. ContactManager requires that all Guidewire core applications implement the `ABClientAPI` interface. Implementing this interface ensures that ContactManager can broadcast contact changes to the core applications by using standard web services, and that the core applications have logic to handle these updates.

ClaimCenter synchronizes contacts as follows:

- If ContactManager sends a contact change, ClaimCenter saves only the contact ID and then uses the contact automatic synchronization mechanism to update all local instances of the contact.

ClaimCenter can have multiple local instances of any contact, one instance for each claim. ClaimCenter uses its automatic synchronization mechanism to ensure that all instances are synchronized. This behavior is different

from the other Guidewire core applications. PolicyCenter and BillingCenter have just one local instance of a linked contact and apply all changes sent from ContactManager to their linked contacts.

- Like the other core applications, ClaimCenter automatically sends updates to ContactManager for linked contacts. In the default configuration, for a non-vendor contact, ClaimCenter calls ContactManager and specifies that the change is to be applied to the ContactManager contact. For a vendor contact, when the ClaimCenter user updates the contact changes, ClaimCenter determines if the user has contact permissions to perform the action. ClaimCenter then calls ContactManager appropriately:
 - If the user has permissions, ClaimCenter makes a call to ContactManager specifying that the vendor contact update is to be applied to the ContactManager contact.
 - If the user does not have permissions, ClaimCenter makes a call to ContactManager specifying that the vendor contact update is to be pending. A pending update has to be reviewed by a ContactManager contact administrator.

Note: For information on defining this vendor contact behavior in the class `ContactSystemApprovalUtil`, see “Linking in ClaimCenter” on page 197.

If the update succeeds, ClaimCenter then updates all local instances of the contact by using the automatic synchronization mechanism. A pending update succeeds if the update is approved in ContactManager and ContactManager notifies ClaimCenter.

- When ContactManager applies a contact change or create that originated from ClaimCenter, ContactManager sends the update to any other Guidewire applications that are integrated with ContactManager. It does not send the update back to ClaimCenter.

Configuring automatic synchronization with ClaimCenter

ClaimCenter contacts that are linked to ContactManager contacts can be synchronized automatically. The synchronizing mechanism ensures data consistency between ClaimCenter and ContactManager. This mechanism updates centrally managed contacts appropriately when a contact changes.

You can configure automatic synchronization in ClaimCenter, as described “ClaimCenter synchronizing support” on page 204.

ClaimCenter synchronizing support

There are a number of ways to configure synchronization of contacts in ClaimCenter. You can use the `AutoSync` attribute, configure rules, and configure the `ContactAutoSync` work queue.

ClaimCenter `AutoSync` attribute of Contact

The ClaimCenter `Contact` entity has a typekey attribute called `AutoSync` that controls whether contacts are automatically synchronized. By default, all contacts are synchronized, an `AutoSync` setting of `Allow`.

For any `Contact` instance, the `AutoSync` value can be one of three codes:

<code>Allow</code>	Allow the contact to be synchronized automatically. In a rule, this value would be <code>TC_ALLOW</code> .
<code>Disallow</code>	Do not allow the contact to be synchronized. In a rule, this value would be <code>TC_DISALLOW</code> .
<code>Suspended</code>	The contact was synchronized in the past, but synchronizing is not currently allowed. In a rule, this value would be <code>TC_SUSPENDED</code> . ClaimCenter typically sets <code>AutoSync</code> to this value when a claim is closed. If the claim is reopened, ClaimCenter sets the value of <code>AutoSync</code> to <code>Allow</code> .

Rules that affect contact synchronization

In the default configurations of ClaimCenter and ContactManager, all contacts are set to allow automatic synchronization. However, for automatic synchronization to run, you must schedule the contact auto sync work queue.

You can determine which contacts you want your users to be able to synchronize. For example, you might not want to synchronize `Person` and `ABPerson` contacts, but you might want to synchronize all `Vendor` and `ABVendor` contacts. Or, you might want to synchronize only contacts that meet particular criteria.

You can use the Preupdate rules in Guidewire Studio™ for ClaimCenter to set automatic synchronization values, determine how addresses update, and set the minimum required tags.

For example, there are predefined **ContactPreupdate** rules in ClaimCenter. To see them, open ClaimCenter Studio. Then in the **Project** window, navigate to **configuration→config→Rule Sets→Preupdate→ContactPreupdate**.

COP01000 - Update Check Address

Updates addresses that have changed for synchronized contacts.

COP02500 - Set all Vendors to auto sync

Ensures that all contacts added to a claim are set to synchronize. For all contacts, AutoSync is set to Allow.

COP03000 - Add default tags

Ensures that the tags for a contact are the minimum required tags for ClaimCenter. The default tags are defined in the class `gw.api.contact.CCContactMinimalTagsImpl`. By default, all contacts get the Claim Party tag, and all vendor types, such as `PersonVendor` and `CompanyVendor` and their subtypes, also get the Vendor tag.

See also

- “Running and scheduling the contact auto sync work queue” on page 205

Setting up the contact auto sync work queue

To process each change to synchronized contacts, ClaimCenter uses a work queue named `CCContactAutoSyncWorkQueue`. You can configure work queues in `work-queue.xml`.

You can open this file in Guidewire Studio™ for ClaimCenter. In the **Project** window, navigate to **configuration→config→workqueue**, and then double-click `work-queue.xml` to open it in the editor.

Note: After you edit any of the configuration files described in this topic, you must rebuild and redeploy ClaimCenter for the changes to take effect.

The following code shows the default settings for this work queue:

```
<work-queue
  workQueueClass="com.guidewire.cc.domain.contact.CCContactAutoSyncWorkQueue"
  progressinterval="600000">
  <worker instances="1" maxpollinterval="0"/>
</work-queue>
```

You can change the attributes for a work queue and add additional worker instances in `work-queue.xml`.

The comments at the beginning of the `work-queue.xml` file document the attributes for a work queue and provide some guidelines for adding worker instances.

See also

- For general information on configuring work queues, see the *System Administration Guide*

Running and scheduling the contact auto sync work queue

You can have your work queues process changes on a schedule or in real time as they happen. The `InstantaneousContactAutoSync` parameter in the ClaimCenter `config.xml` file controls this behavior. This parameter determines how ClaimCenter updates copies of a contact instance that has changed, either by a user action in ClaimCenter or by a change notification from ContactManager.

To open this file, start Guidewire Studio™ for ClaimCenter. Then, in the **Project** window, navigate to **configuration→config** and double-click `config.xml`. After changing this parameter, you must rebuild and redeploy ClaimCenter for the changes to take effect.

By default, the `InstantaneousContactAutoSync` parameter is set to `true`:

```
<param name="InstantaneousContactAutoSync" value="true"/>
```

There are also two configuration parameters in `config.xml` that can improve performance of contact synchronization:

ContactAutoSyncWorkItemChunkSize

Specifies the maximum number of contacts linked to a single `AddressBookUID` that each `ContactAutoSync` work item is to process. The default value is 400.

ContactAutoSyncBundleCommitSize

Specifies the maximum number of contacts that match a single `AddressBookUID` in each bundle commit. The default value is 15.

For example, if 1600 `ClaimCenter` local contacts are linked to a single `ContactManager` contact (one `AddressBookUID` for all 1600 contacts), then with these parameters set to their default values:

- There are four work items with 400 contacts each.
- Each work item does 27 commits (400/15).

IMPORTANT With `InstantaneousContactAutoSync` set to `true`, updating local instances of a contact can affect system performance if you have a large number of local instances. Additionally, if `ClaimCenter` receives a contact change notification from `ContactManager`, `ClaimCenter` updates all local instances of the contact, even instances open for edit. If a `ClaimCenter` user is editing a local instance of that contact, the user will be unable to save the changes made in the editing session. The user will have to exit the editing session and start over with the newly updated contact data received from `ContactManager`.

Setting `InstantaneousContactAutoSync` to `false` causes `ClaimCenter` to perform synchronized updates on the schedule set for the `ContactAutoSync` work queue in the `scheduler-config.xml` file. To open this file in the editor, start Guidewire Studio™ for `ClaimCenter`, and in the **Projects** window, navigate to **configuration**→**config**→**scheduler** and double-click `scheduler-config.xml`. You must rebuild and redeploy `ClaimCenter` for changes to take effect.

By default, `ContactAutoSync` is set to run at 5:00 am and 5:00 pm every day. It is on this schedule to ensure that all contacts are synchronized before the `Financials Escalation` work queue, `FinancialsEsc`, runs every day at 6:05 am and 6:05 pm. Synchronizing contacts before running the `Financials Escalation` work queue prevents `Financials Escalation` from having validation errors for contacts that are not synchronized. If you change the `ContactAutoSync` times, also schedule `Financials Escalation` to run at an appropriate time afterward.

The default `ContactAutoSync` setting looks like this:

```
<!-- Contact Auto Sync batch process should run every day at 5 am and 5 pm.
      It needs to run before the financials escalations to prevent those
      from triggering validation errors from out of sync contacts.-->
<ProcessSchedule process="ContactAutoSync">
  <CronSchedule hours="5,17"/>
</ProcessSchedule-->
```

- If you set `InstantaneousContactAutoSync` to `false` and do not schedule `ContactAutoSync`, the work queue does not run and the system does not process changes.
- If you set `InstantaneousContactAutoSync` to `true` and you have `ContactAutoSync` scheduled, changes process immediately, and then the scheduled `ContactAutoSync` work queue also runs as scheduled.

See also

- For general information on administering and configuring scheduled work queues with the work queue scheduler, see the *System Administration Guide*.

Initiating a manual synchronization from ClaimCenter

You can initiate a contact synchronization in the `ClaimCenter` user interface. Centrally managed contacts have a status message that informs you when they are not synchronized. For example, you have a claim open and on the **Parties Involved**→**Contacts** screen, the **Basics** card for the contact says:

This contact is linked to the Address Book but is out of sync

When you see this status message, you can click **Copy from Address Book** to synchronize the Address Book data for this one contact. Clicking this button copies the contact data from ContactManager to ClaimCenter.

Synchronizing ClaimCenter contact fields

When ClaimCenter synchronizes contact data with ContactManager, it updates the fields in the ClaimCenter contact with the centralized ContactManager data.

The system overwrites the fields defined in the ClaimCenter `ContactMapper` class. This class also defines the fields that ClaimCenter sends to ContactManager. Use this class to control which fields are synchronized between ClaimCenter and ContactManager.

In addition, you can edit the `gw.plugin.contact.ab1000.RelationshipSyncConfig` class in Guidewire Studio™ for ClaimCenter to specify the contact relationships to include or exclude in a synchronization.

Excluding properties from contact synchronization

In general, to determine which properties to consider in determining if a contact is synchronized, in most cases you use the `ContactMapper` class. With the exception of contact relationships, `ContactMapper` must list all contact fields to be transferred between ClaimCenter and ContactManager. If you omit a property from this class, it is not transferred or considered for synchronization.

By default, all properties listed in `ContactMapper` are considered in determining synchronization status.

If there is a property in `ContactMapper` that you want excluded from the synchronization check, use the `withAffectsSync` method and set its parameter to `false`. For example, the following code excludes the property `CategoryScores`:

```
arrayMapping(Contact#CategoryScores)
    .withMappingDirection(TO_BEAN)
    .withAffectsSync(false),
```

See also

- “Excluding contact fields from ClaimCenter contact synchronization” on page 282

Including and excluding contact relationships in synchronization

To include or exclude a contact relationship in the contact synchronization, use the `includeRelationship` and `excludeRelationship` methods of the ClaimCenter `RelationshipSyncConfig` class.

IMPORTANT Typically, you include only relationships that appear on the **Contact Basics** panel of the ClaimCenter **Parties Involved** screen. In particular, avoid zero-or-more relationship types. These types can grow quickly and can result in performance issues as the system pulls down many contacts from ContactManager.

You open the `RelationshipSyncConfig` class for editing in Guidewire Studio™ for ClaimCenter.

The `RelationshipSyncConfig` class has two methods that support including or excluding relationships when synchronizing a Contact subtype:

includeRelationship

Specifies relationships to include in synchronizing the specified Contact or Contact subtype.

excludeRelationship

Specifies relationships to exclude in synchronizing the specified Contact or Contact subtype.

These two methods have the same parameters:

contactType

The Contact subtype for which the synchronizable relationship will be added or excluded.

contactBidiRel

The relationship `ContactBidiRel`, a valid typecode in the `ContactRel` typelist. The typecode must be specified as an enumerated typelist value, such as `TC_EMPLOYER` or `TC_PRIMARYCONTACT`.

appliesToSubtype

If `true`, the relationship exclusion or inclusion applies to all subtypes of `contactType` unless overridden by an `includeRelationship` or `excludeRelationship` call. If `false`, the method applies only to the `Contact` subtype specified in `contactType`.

Note: You use a coding pattern called *method chaining* to add the method calls to the `init` method of the `RelationshipSyncConfig` class. With method chaining, the object returned by one method becomes the object from which you call the next method in the chain.

The `init` method can have multiple `includeRelationship` or `excludeRelationship` method calls chained to it. Add the methods before the final `create` method in the chain.

For example, you might include a relationship for one contact type but exclude it for its child subtypes. The following method call, which is included in the `RelationshipSyncConfig` class in the base configuration, includes the guardian relationship for a `Person` entity, but not for its subtypes:

```
.includeRelationship(Person, TC_GUARDIAN, false )
```

The default behavior for an entity's relationships is that the system ignores them. Only in certain cases do you need to use `excludeRelationship`, such as overriding a setting of `includeRelationship` in a parent type. For example, the default `Contact` definition includes the relationship `primarycontact` for all `Contact` entities and subentities. However, the `Person` definition overrides the `Contact` definition and excludes the `primarycontact` relationship only for `Person` entities, as shown in the following code snippet:

```
return new RelationshipSyncConfigBuilder<RelationshipSyncConfig>(
    new RelationshipSyncConfig())
    .includeRelationship(Contact, TC_PRIMARYCONTACT, true)
    .excludeRelationship(Person, TC_PRIMARYCONTACT, false)
    <!-- ... -->
    .create()
```

See also

- “Linking a contact” on page 195.
- For information on method chaining, see the topic on chaining query builder methods in the *Gosu Reference Guide*.

ContactManager rules

ContactManager provides rules to handle business processes. These rules use the same Guidewire Studio™ editors and infrastructure described in the *Rules Guide*.

ContactManager rule sets

This topic describes the sample rules that Guidewire provides as part of the base ContactManager application. You access these rule sets in Guidewire Studio™ for ContactManager by navigating in the **Project** window to **configuration→config→Rule Sets**.

ContactManager provides the following rule set categories in the base product:

Rule set category	Description
EventMessage	Rules that handle communication with integrated external applications. See “EventMessage rule set category” on page 209. For information on events and event messaging, see the <i>Integration Guide</i> .
Preupdate	Rules triggered any time that a contact or other high-level entity changes. See “Preupdate rule set category” on page 211.
Validation	Rules that check for missing information or invalid data on ABContact and Region entities. See “Validation rule set category” on page 213.

EventMessage rule set category

In the base configuration, there is a single rule set, **EventFired**, in the **EventMessage** rule category. The rules in this rule set:

- Perform event processing.
- Generate messages about events that have occurred.

ContactManager calls the Event Fired rules if an entity involved in a bundle commit triggers an event for which a message destination has registered interest. As part of the event processing, ContactManager:

- Runs the rules in the Event Fired rule set once for every event for which a message destination has registered interest.
- Runs the Event Fired rule set once for each destination that is listening for that particular event. It is possible for ContactManager to run the Event Fired rule sets multiple times for each event, once for each destination interested in that event.

Messaging events

ContactManager automatically generates certain events, called *standard* events, for most top-level objects. In general, events are generated for any addition, modification, removal, or retirement of a top-level entity.

For example, ContactManager automatically generates the following events on the ABContact object:

- ABContactAdded
- ABContactChanged
- ABContactRemoved
- ABContactResync
- ABContactPendingChangeRejected

It is also possible to create a custom event on an entity by using the `addEvent` method. This method takes a single parameter, *eventName*, which is a `String` value that sets the name of the event:

```
entity.addEvent(eventName)
```

See also

- “ContactManager messaging events” on page 269
- For information on messaging events and custom events, see the *Integration Guide*

Link a message event to a message destination

About this task

You can use the Messaging editor to link an event to a message destination.

- A *message destination* is an external system to which it is possible to send messages.
- A *message event* is an abstract notification of a change in ContactManager that is of possible interest to an external system. For example, events can be adding, changing, or removing a Guidewire entity.

Procedure

1. At a command prompt, navigate to the ContactManager installation folder, and then start Guidewire Studio™ for ContactManager by entering the following command:

```
gwb studio
```

2. In the **Project** window, navigate to **configuration→config→Messaging**, and then double-click `messaging-config.xml`.

In the Messaging editor, you can associate one or more events with a particular message destination.

See also

- For information on using the Messaging editor, see the *Configuration Guide*
- For information on message destinations, implementing messaging plugins, and messaging and events, see the *Integration Guide*

Message destinations and message plugins

Each message destination encapsulates all the necessary behavior for an external system, but uses three different plugin interfaces to implement the destination. Each of the following plugins handles different parts of what a destination does:

- The message request plugin handles message preprocessing.
- The message transport plugin handles message transport.
- The message reply plugin handles message replies.

You register new messaging plugins first in the Plugins editor in Guidewire Studio™. After you create a new implementation, Studio prompts you for a plugin interface name, and, in some cases, a plugin name. Use that plugin

name in the Messaging editor in Studio to register each destination. You must register your plugin in two different editors in Studio, first in the Plugins registry and then in the Messaging editor.

IMPORTANT After the ContactManager application server starts, ContactManager initializes all message destinations. ContactManager saves a list of events for which each destination requested notifications. Because this initialization happens at system startup, you must restart the ContactManager application if you change the list of events or destinations.

See also

- For information on using the messaging editor, see the *Configuration Guide*
- For information on implementing messaging plugins, see the *Integration Guide*

Generating messages

Use the method `createMessage` in rules to create message text. The message that the rule creates can be either a simple text message or a more involved, constructed message.

The following code is an example of a simple text message that uses the Gosu in-line dynamic template functionality to construct the message. Gosu in-line dynamic templates combine static text with values from variables or other calculations that Gosu evaluates at run time. For example, the following `createMessage` method call creates a message that lists the event name and the entity that triggered this rule set.

```
messageContext.createMessage("${messageContext.EventName} - ${messageContext.Root}")
```

See also

- For information on generating new messages in event-fired rules, see the *Integration Guide*

Modifying entity data in event fired rules and messaging plugins

Be careful about modifying entity data in Event Fired rules and messaging plugin implementations. Use these rules to perform only the minimal data changes necessary for integration code. Entity changes in these code locations do not cause the application to run or rerun validation or preupdate rules. Therefore, do not change fields that might require those rules to run again. Only change fields that are not modifiable from the user interface. For example, you might set custom data model extension flags only used by messaging code.

Important message configuration caveats

ContactManager does not support the following:

- Adding or deleting business entities from Event Fired rules or messaging plugins, even indirectly through other APIs.
- Calling any message acknowledgment or any skipping methods such as the `Message` methods `reportAck`, `reportError`, or `skip` in rules. Use those methods only in messaging plugins. This prohibition also applies to Event Fired rule set execution.
- Creating messages outside the Event Message rule set.

See Also

- For an overview of messaging, see the *Integration Guide*
- For information on using the Messaging editor, see the *Configuration Guide*

Preupdate rule set category

Use the preupdate rule sets to perform domain logic or validation that must be committed before the entity in question is committed. Only a change to an entity that implements the `Validatable` delegate can trigger a preupdate rule.

Typically, these rules execute before the validation rule set. Preupdate rules can perform a wide variety of actions as needed. The primary use of the preupdate rules in the base configuration is to keep a history of changes to contacts and addresses.

Additionally, there is a Pending Contact Change Preupdate rule set with rules that handle rejection of either a pending contact creation or a pending contact update.

See also

- For information on how the validation and preupdate rules work together in performing validation, see “Overview of ContactManager preupdate and validation” on page 213.

Entities that trigger preupdate rules

The following entities trigger preupdate rules in the ContactManager base configuration:

- ABContactContact
- ABContact
- Address
- PendingContactChange

For an entity to trigger the preupdate rules, it must implement the `Validatable` delegate. All entities that implement the `Validatable` delegate trigger the validation rules as well.

ContactManager runs the preupdate and validation rules if:

- An instance of a validatable entity is created, modified, removed, or retired.
- A subentity of a validatable entity is created, modified, removed, or retired, and the validatable entity is connected to the subentity through a `triggersValidation="true"` link.

IMPORTANT In running the preupdate rule set, ContactManager first computes the set of objects on which to run the preupdate rules. It then runs the rules on this set of objects. If a preupdate rule modifies an entity, the preupdate rules for that entity do not fire unless the schedule for preupdate rules already includes this entity. In other words, changing an object in a preupdate rule does not cause the preupdate rules to run on that object as well. Additionally, creating a new entity in a preupdate rule does not cause the preupdate rules to run on that entity.

Creating preupdate rules for custom entities

You can create preupdate rules for entities that you create—entities that are not part of the base Guidewire ContactManager configuration.

- The entity must implement the `Validatable` delegate with an `implementsEntity` element that has its **Name** set to `Validatable`.
- The preupdate and validation rule sets that you want the custom entity to trigger must conform to the following naming convention:
 - Put your preupdate rules in the **Preupdate** rule set category and name the rule set `entity-namePreupdate`.
 - Put your validation rules in the **Validation** rule set category and name the rule set `entity-nameValidationRules`.

You must create a rule set category with the same name as the extension entity and add the word **Preupdate** to it as a suffix. You then put your rules in this rule set.

For example, if you create an extension entity named `NewEntityExt`, you need to create a rule set to hold your preupdate rules and name it `NewEntityExtPreupdate`.

See also

- For information on creating new rule sets, see the *Rules Guide*.

ABContactContact preupdate rule set

Use the ABContactContact preupdate rules to modify ABContactContact and related entities. This rule set in the base configuration has rules that track the history on changes to a contact's related contacts.

ABContact preupdate rule set

Use the ABContact preupdate rules to modify ABContact and related entities. This rule set in the base configuration has rules that:

- Set the BatchGeocode field for a vendor contact.
- Track history for creation of a contact, removal of a related contact, and changes to a contact's name, phone numbers, addresses, and so on.
- For an ABContact entity that has changed, remove entries for that entity from the cache for the **Pending Changes** screen.

See also

- “Pending changes screen cache” on page 255

Address preupdate rule set

Use the Address preupdate rules to modify Address and related entities. This rule set in the base configuration has rules that track the history on changes to a contact's primary and secondary addresses.

PendingContactChangePreupdate rule set

The pending contact change preupdate rules handle rejection of either a pending contact creation or a pending contact update. Pending contacts are reviewed by a contact administrator and can be either accepted or rejected. Rejected changes trigger an event that sends a message to the core application, which notifies the user who made the change that the change was rejected.

Additionally, for a rejected pending change to an ABContact entity, there is a rule that removes entries for that entity from the cache for the **Pending Changes** screen.

See also

- “Review pending changes to contacts” on page 253
- “Pending changes screen cache” on page 255

Validation rule set category

ContactManager uses validation rules to ensure that:

- A region zone is available in the zone lookup for the organization.
- The date of birth of each ABContact instance is in the past.
- The country code portion of all non-null phone numbers for each ABContact instance is correctly formatted.

Overview of ContactManager preupdate and validation

ContactManager runs preupdate and validation rules every time it does a *database bundle commit*—commits data to the database. The validation rules execute after ContactManager runs all preupdate callbacks and preupdate rules. ContactManager runs the validation rules as the last step before writing data to the database.

Before applying rules during the bundle commit operation, ContactManager builds a validation object graph according to configuration settings. The graph contains possible targets for rule execution and is used by both the preupdate and the validation rules.

Preupdate rules, unlike validation rules, can modify additional objects during execution. Therefore, there can be additional objects that need to be constructed after the preupdate rules run and before the validation rules run.

Following are some examples indicating that the set of entities to be validated can be a super set of the entities for which preupdate rules are run:

Preupdate rules modify an entity that is not in the commit bundle

ContactManager can run additional validation rules if the entity being modified also triggers validation.

Preupdate rules modify only entities that are in the commit bundle

ContactManager runs the validation rules for the same set of entities on which the preupdate rules ran.

Preupdate rules modify an entity that triggers validation for one of the top-level entities

There is no difference from the top-level entity validation rules because they were already going to run.

Note: Preupdate rules are not recursive. They do not run a second time on objects modified during execution of the preupdate rules. ContactManager just adds any objects modified by the preupdate rules to the list of objects needing validation, as described by the validation graph.

Overview of ContactManager validation

For an entity to have preupdate or validation rules associated with it, the entity must be validatable. To be validatable, the entity must implement the `Validatable` delegate. In the base configuration, ContactManager comes preconfigured with the following high-level entities that can trigger validation:

- `ABContact`
- `ABContactContact`
- `Activity`
- `Address`
- `Group`
- `PendingContactChange`
- `Region`
- `User`

ContactManager can validate these entities in no particular order.

[See also](#)

- “Top-level ContactManager entities that trigger validation” on page 215
- For information on `<implementsEntity>`, see the *Configuration Guide*

Validation graph in ContactManager

During database commit, ContactManager performs validation on the following entities:

- Any validatable entity that is itself either updated or inserted.
- Any validatable entity that refers to an entity that has been updated, inserted, or removed.

ContactManager gathers all the entities that reference a changed entity into a virtual graph. This graph maps all paths from each type of entity to the top-level validatable entity, such as `ABContact`. These paths are queried in the database or in memory to determine which validatable entities, if any, refer to the entity that was inserted, updated, or removed.

ContactManager determines the validation graph by traversing the set of foreign keys and arrays that trigger validation. For example, suppose that the data model marks the `Address` array on `ABContact` as triggering validation. Therefore, any changes made to an address causes the Rules engine to validate the contact as well.

ContactManager follows foreign keys and arrays that trigger validation through any links and arrays on the referenced entities down the object tree. For example, you might end up with a path like `ABContact` → `ABContactAddress` → `Address`. To actually trigger validation, each link in the chain—`Address` and `ABContactAddress`—must be marked as triggering validation, and `ABContact` must implement the `Validatable` delegate.

ContactManager stores this path in reverse order in the validation graph. Thus, if an address changes, ContactManager follows the path to find the contact that references a changed address. ContactManager transverses the tree from address to contact address, and finally to the contact—Address → ABContactAddress → ABContact.

Top-level ContactManager entities that trigger validation

To be validatable, an entity, subtype, delegate, or base object must implement the `Validatable` delegate. If an entity has a foreign key or array for which `triggersValidation` is `true`, the referenced entities must also implement the `Validatable` delegate.

The following table lists the entities that trigger validation in the base ContactManager configuration.

Validatable entity	Foreign key	Array	Comments
ABContact	PrimaryAddress	<ul style="list-style-type: none">ContactAddressesSourceRelatedContactsTargetRelatedContacts	In the base configuration, ContactManager runs the preupdate and validation rules on the ABContact object during database commit if any objects with <code>triggersValidation="true"</code> have been modified.
User	<ul style="list-style-type: none">ContactCredentialUserSettings	Attributes	In the base configuration, ContactManager does not have any preupdate or validation rules for the User entity.

Configuring personal data destruction in ContactManager

ContactManager supports destruction of some kinds of data. Destruction can mean either purging the data completely from the database or it can mean obfuscating data, making the original contents permanently unreadable. Guidewire recognizes the need for insurers to be able to destroy personal information both on an on-demand basis or on a time-based basis. Destruction can be mandated by regulation or business practices, within the requirements of regulation, codes of conduct, or other business practices.

Overview of data destruction

Note: The data destruction features described in these topics provide a set of features that help enable insurers to comply with some of their data destruction requirements. These requirements may be driven by insurers' policies and practices, as well as by their interpretation of various regulatory requirements. Such regulatory requirements may come from, for example, the European Union General Data Protection Regulation (GDPR) or the New York State Cybersecurity Requirements for Financial Services Companies law.

Data destruction is the process of requesting that data be destroyed, making the data impossible to retrieve. Data destruction is typically initiated with a request that specifies a contact or user whose data is to be destroyed. In the base configuration, ContactManager provides a web service that is intended to be called by an external application. You use the external application to manage the destruction of the data across Guidewire applications.

Data destruction can be implemented as either purging or obfuscation of data, depending on the data to be destroyed.

Purging is a form of data destruction that completely removes contact data from ContactManager. There can be multiple objects associated with the contact that are also removed as they are detected by traversing the entity domain graph.

Obfuscation is a form of data destruction that permanently overwrites fields, such as user contact fields, with data that replaces the original data. Some actual removal of data can also be involved, such as deletion of an address referenced only by one user.

Obfuscation might be required if destroying the data affects contacts that cannot be destroyed. For example, purging user data for a former employee could affect hundreds or even thousands of contacts. Therefore it makes more sense to obfuscate the data for the user and leave the other data alone.

Encapsulation of business logic for retention and destruction

Regulations, codes of conduct, and other generally accepted business practices vary from jurisdiction to jurisdiction. Additionally, business policies and interpretation of conflicting legal requirements vary from insurer to insurer.

Therefore no single approach meets the needs of all insurers. To accommodate varying needs, ContactManager provides a configurable solution that captures business logic for retention and destruction in one place.

There is a configurable plugin that has access to the business objects to be removed through the `ABContact` root object. The examination of objects to be removed starts with the root object and traverses a graph of objects, enabling detailed examination of the business objects. You can mark requests requiring user review—manual intervention—for those data destruction requests that require special handling, prior to the destruction actually occurring.

See also

- “ContactManager entity domain graph” on page 227

Notification of data protection officer on errors or conflicts

Requirements for destruction and for retention can conflict with each other. While the plugin class might be able to resolve conflicts in a generic way, situations can arise when the two sets of requirements are not reconcilable. Additionally, the data destruction process can encounter errors. In these situations, notification is done through a configurable plugin.

The default behavior of this plugin is that a message is logged that describes the situation.

After the situation has been resolved, the destruction request can be queued again for reprocessing.

See also

- “ContactManager Data Protection Officer” on page 230

Wide-swath data destruction

In many situations, there is a need to destroy the personal data related to a specific business object, a contact. Specifically, the `ABContact` entity and its subentities can require this kind of destruction.

This object can affect many individual data objects. A single call allows the entirety of related data to be removed. In the case where these business objects are nested, a best-effort destruction is performed.

ContactManager components provide the ability to purge rows from the database for business objects such as `ABContact` and related data. This approach is suitable for high-volume data destruction.

See also

- “Data destruction purge configuration” on page 231

Individual-entity data destruction

While wide-swath data destruction meets the needs of the insurer in most cases, there are special cases in which specific personal data cannot be deleted. For example, there might be database integrity concerns, or the data to be deleted, such as data for previous employee, might be related to a large number of contacts.

In such cases, where individual instances of data cannot be deleted, ContactManager provides the ability to obfuscate data. Obfuscation can include wiping a field completely, replacing it with a neutral value, or replacing it with a unique, irreversible value. Additionally, some actual removal of data can also be involved, such as deletion of an address referenced only by one user.

The entities and fields to which obfuscation can be applied, as well as the method for determining the replacement value, are configurable.

See also

- “Data obfuscation in ContactManager” on page 234

Integration with Guidewire core applications

ContactManager coordinates with Guidewire core applications on any request to destroy an `ABContact` object or a subobject of `ABContact`. This coordination is the same as with any contact removal. ContactManager cannot purge an `ABContact` until it receives permission from any core applications that are installed.

However, `User` objects are local to ContactManager, so obfuscation of a user can proceed without consulting core applications.

See also

- “Data destruction purge configuration” on page 231
- “Data obfuscation in ContactManager” on page 234

Integration with other systems

ContactManager needs to be able to respond to data destruction requests from external systems, as well as have the ability to notify data consumers of data destruction.

ContactManager provides a web service that:

- Takes a reference to an individual contact.
- Takes application-specific action to destroy the data related to that contact.
- Reports back to the caller on the level of success of the request. Callers can query the status of a given request.

See also

- “Data destruction web service” on page 219

Notification of downstream systems

ContactManager provides a messaging system to assist you in ensuring that the destruction of personal data flows into systems connected with components. Additionally, you might need to notify outside organizations that process data on your behalf. The messaging system supports broadcasting personal data destruction response messages.

These messages are delivered by using the existing ContactManager guaranteed-delivery messaging system.

See also

- “PersonalDataPurge event” on page 232

Data destruction configuration parameters

In the base configuration of ContactManager, data destruction is not enabled. You must set the following configuration parameters in `config.xml` to enable data destruction:

- `PersonalDataDestructionEnabled` – Set this parameter to `true` to enable destruction of personal data. In the base configuration, this parameter is `false`.
- `ContactDestructionRequestAgeForPurgingResults` – Used by the `RemoveOldContactDestructionRequest` work queue to determine the age of `PersonalDataDestructionRequest` objects that have a status of `Finished`. In the base configuration, this parameter is set to 10 days.

Data destruction web service

ContactManager provides a web service that enables external software programs to initiate and track requests to destroy data. In the base configuration, this web service, `PersonalDataDestructionAPI`, enables an external application to request:

- Destruction of a contact’s data by `AddressBookUID` or by `PublicID`.
- Destruction of a user by user name.

In the base configuration, the `PersonalDataDestroyer` obtained by the class that implements the `PersonalDataDestructionPlugin` uses an `ABContact` `PublicID`. You can configure the `PublicID` to correspond to an entity other than `ABContact`.

The requests for removal return a unique `requesterID` that can be used to check the status of the request. Additionally, this `requesterID` is available in the plugin notification call when the request has been completed.

Note: The `PersonalDataDestructionAPI` checks for both retired and active contacts when a contact data destruction request is received. When implementing data obfuscation for contacts, you must evaluate the need to look for related objects that are retired in the database. Retired objects can be obfuscated in the same fashion as active objects, but must be retrieved from the database with a query that is specified to look for retired objects. For example: `query.withFindRetired(true)`.

IMPORTANT The external software program that calls the web service must destroy data for a contact in the core applications before requesting that `ContactManager` destroy the contact. If you have more than one core application installed, the external application must request that the contact be destroyed in all of them before sending the request to `ContactManager`. When `ContactManager` processes a request to destroy a contact, it first verifies with all installed core applications that it can destroy the contact. If any core application indicates that the contact cannot be destroyed, `ContactManager` does not proceed with the destruction request and notifies the web service that the contact cannot be destroyed.

See also

- “Defining the Destroyer” on page 231
- “Specifying which objects in the graph can be destroyed” on page 233

PersonalDataDestructionAPI web service methods

`PersonalDataDestructionAPI` provides the following methods:

- `requestContactRemovalWithABUID(addressBookUID : String, requesterID: String) : String` – A request to destroy a contact by `AddressBookUID`. In `ContactManager`, the `AddressBookUID` is checked against the `LinkID` of an `ABContact` object. This method is implemented as an asynchronous process that uses work queues.
- `requestContactRemovalWithPublicID(contactPublicID : String, requesterID: String) : String` – A request to destroy a contact by `PublicID`. In `ContactManager`, the contact is an `ABContact`. This method is implemented as an asynchronous process that uses work queues.
- `doesABUIDExist(addressBookUID: String): boolean` – Uses `translateABUIDToPublicIDs` as defined in the class that implements the `PersonalContactDestroyer` interface. In `ContactManager`, this translation method compares the `AddressBookUID` to the `LinkID` of an `ABContact`.
- `doesContactWithPublicIDExist(publicID: String): boolean` – In `ContactManager`, the contact is an `ABContact` object.
- `currentDestructionRequestStatusByRequestID(uniqueRequestID : String): DestructionRequestStatus`
- `destroyUser(username : String) : boolean` – Given a `ContactManager` user name, verifies the existence of the credential and the user. This method is a synchronous destruction request that does not involve work queues and does not require coordination with core applications. It works with the `Contact` object and not with `ABContact` because `User` is a subtype of `Contact`.
 - If the credential exists and the user does not, then the method obfuscates the credential, logs that the user does not exist, and returns `credential.IsObfuscated`.
 - If both credential and user exist, the method attempts to obfuscate the user, which obfuscates both the credential and the `UserContact` object. If the obfuscation succeeds, the method returns `true`. If not, it logs that there was a failure and returns `false`.

See also

- “Defining the Destroyer” on page 231

Lifecycle of a personal data destruction request

The lifecycle of a contact removal request depends on the method that the external system calls to start the request. The lifecycle, also called an *asynchronous* personal data destruction request, is started by a call either to `requestContactRemovalWithABUID` or `requestContactRemovalWithPublicID`. For these two web service method calls, the external system has either the `AddressBookUID` or the `PublicID` of the contact whose data to be destroyed. The destroy action performed is defined in the `ContactManager` plugin class that implements the `PersonalDataDestruction` plugin interface.

Note: In the base configuration, the `destroyUser` method is synchronous and initiates obfuscation. It does not use work queues, and therefore does not participate in the personal data destruction request lifecycle. Additionally, because users of the application are subtentities of `Contact`, this method works with `Contact` objects and not `ABContact` objects.

If the web service determines that the request is an existing one, it adds the specified `requesterID` value to the existing destruction request and does not start a new request.

If the web service determines that the request is a new one, the web service:

1. Does the following depending on whether the request is for an `AddressBookUID` or `PublicID`:
 - If the web service call was to `requestContactRemovalWithABUID`, the web service:
 - Creates a `PersonalDataDestructionRequest` object for the `LinkID` of the `ABContact`.
 - Adds a `PersonalDataContactDestructionRequest` object for the related `PublicID` value, obtained from a call to the `PersonalDataDestroyer` implementation.
 - If the web service call was to `requestContactRemovalWithPublicID`, the web service creates a `PersonalDataContactDestructionRequest` object for the `PublicID` of the `ABContact`.
2. Adds a `PersonalDataDestructionRequester` object using `requesterID`.
3. The `DestroyContactForPersonalData` work queue checks for requests in the `ReadyToAttemptDestruction` category, status `New` or `ReRun`, and calls the `Destroyer`.

The class `PersonalDataContactDestructionWorkQueue`, which implements this work queue, calls the following method:

```
PersonalDataDestructionController.destroyContact(contactPurgeRequest)
```

- If the request status is in the `DestructionStatusFinished` category, the queue marks the date of destruction for the contact destruction request.
 - If the request status is `ManualInterventionRequired`, you must implement code that notifies the data protection officer. That user must determine what to do and then set the status to `ReRun` so the `DestroyContactForPersonalData` work queue can run it again.
4. The `NotifyExternalSystemForPersonalData` work queue looks at all `PersonalDataContactDestructionRequest` objects that are associated with a `PersonalDataDestructionRequest`. If they all have a status that is in the `DestructionStatusFinished` category, the work queue does the notification.
 5. The `NotifyExternalSystemForPersonalData` work queue notifies the external system by using `PersonalDataDestructionRequester` objects. As part of this notification, the work queue calls the `PersonalDataDestruction` plugin method `notifyExternalSystemsRequestProcessed`.
 6. The `RemoveOldContactDestructionRequest` work queue removes all requests that satisfy both of the following criteria:
 - The date obtained by adding the value of the configuration parameter `ContactDestructionRequestAgeForPurgingResults` to the value of `PersonalDataContactDestructionRequest.purgedDate` is less than or equal to today's date.
 - The `PersonalDataContactDestructionRequest` object has a typecode that is in the `DestructionStatusFinished` category.

Personal data destruction request entities

Three kinds of entities are created when a request is made to purge a contact:

PersonalDataDestructionRequest

Holds the AddressBookUID (LinkID) and information regarding the parts and result of this destruction request.

PersonalDataContactDestructionRequest

Holds the PublicID of the ABContact and its current destruction status.

PersonalDataDestructionRequester

Holds the external system RequesterID that requested the purge and a unique ID associated with the request assigned by ContactManager.

Example of a request made with AddressBookUID

If the call is made to the web service method `requestContactRemovalWithABUID`, only one contact can have the LinkID. The destruction request causes the following instances to be created:

- One `PersonalDataDestructionRequest`
- One `PersonalDataContactDestructionRequest` object for the PublicID linked to the AddressBookUID request
- One `PersonalDataDestructionRequester`

If a LinkID corresponding to the AddressBookUID is not found, an exception is thrown.

If an existing destruction request for AddressBookUID has `AllRequestsFulfilled` equal to true, then the external system is notified that destruction has already finished.

Example of a request made with PublicID

If two calls are made to the web service method `requestContactRemovalWithPublicID` for the same PublicID, the destruction request creates:

- One `PersonalDataDestructionRequest`
- One `PersonalDataContactDestructionRequest`
- Two `PersonalDataDestructionRequester` objects.

If the PublicID is not found, an exception is thrown.

If an existing destruction request with PublicID has `AllRequestsFulfilled` on the `PersonalDataDestructionRequest` equal to true, then the external system is notified that destruction has already finished.

Typelists for status of personal destruction workflow

The personal destruction workflow uses typecodes to indicate various statuses. These typecodes are defined in three typelists, `ContactDestructionStatus`, `DestructionRequestStatus`, and `ContactDestructionStatusCategory`.

ContactDestructionStatus

When a new contact destruction request is started, the initial status of the destruction object is New. These status values are defined in the `ContactDestructionStatus` typelist. After a destruction attempt is made on the contact, the destroyer is expected to return a status corresponding to how much it was able to destroy:

- **New** – The initial status of the destruction object when a new contact destruction request is started.
- **NotDestroyed** – Nothing could be destroyed.
- **Partial** – Some data was destroyed.
- **Completed** – Everything requested was destroyed.
- **ManualIntervention** – There was an error. This status enables inspection by the Data Protection Officer and must be set before setting `ReRun`.

In the base configuration, `ContactManager` provides code that notifies the Data Protection Officer in the plugin class that implements `PersonalDataDestruction`. Additionally, after the Data Protection Officer takes action, the method `PersonalDataDestructionController.requestContactRemovalRequestWithPublicID` must be

called. You must configure a way to make that method call, such as a button in the ContactManager user interface.

- **ReRun** – Enables another attempt at destruction.

DestructionRequestStatus

You can retrieve the status of the entire destruction request through the `Status` property on the request itself. The status values are defined in the `DestructionRequestStatus` typelist. The general status of the entire destruction request can be:

- **DoesNotExist** – The object to be destroyed does not exist.
- **Unprocessed** – Everything is still in the `New` status.
- **InProgress** – All other combinations of contact destruction statuses.
- **Finished** – Everything is in a final state.

ContactDestructionStatusCategory

Every `ContactDestructionStatus` typecode except `ManualInterventionRequired` has one or more categories.

- **DestructionStatusNotProcessed** – Indicates that the request has not gone through the destruction process.
- **ReadyToAttemptDestruction** – Labels the contact purge request as being ready to be sent to the destroyer.
- **DestructionStatusFinished** – Indicates that the request has finished the destruction process.
- **ReadyToBeNotified** – Labels the request as ready to notify to the external system.

`New` and `ReRun` are under the category `ReadyToAttemptDestruction`.

`New` also is included in the category `DestructionStatusNotProcessed`.

`Partial`, `NotDestroyed`, and `Completed` are under both the category `DestructionStatusFinished` and the category `ReadyToBeNotified`.

Work queues used in personal data destruction

The following work queues are used in personal data destruction: `DestroyContactForPersonalData`, `NotifyExternalSystemForPersonalData`, and `RemoveOldContactDestructionRequest`.

DestroyContactForPersonalData work queue

This work queue finds all `PersonalDataContactDestructionRequest` objects that have a status set to `New` or `ReRun` (category `ReadyToAttemptDestruction`). How far the destruction process went for the found contacts is determined by the `ContactDestructionStatus` returned from the `Destroyer`, the class that implements the `PersonalDataDestroyer` interface.

The contact destruction status is set to the returned status. If the status is `Completed`, `Partial`, or `NotDestroyed` (category `DestructionStatusFinished`), the date of completion is also populated.

An exception is thrown if return status is `New` or if you try to change the status from a typecode in the `DestructionStatusFinished` category.

Note: The class that implements this work queue is `PersonalDataContactDestructionWorkQueue`.

NotifyExternalSystemForPersonalData work queue

This work queue finds all `PersonalDataDestructionRequest` objects that have a status typecode in the `DestructionStatusFinished` category and that have `RequestersNotified` set to `false`. The work queue processes found requests by sending a notification to all associated requesters, and then the work queue marks `RequestersNotified` `true`. If the notification fails, an exception is thrown and `RequestersNotified` remains `false`.

Note: The class that implements this work queue is `PersonalDataDestructionNotifyExternalSystemsWorkQueue`. In your implementation, you must verify that the notification was successful before marking `RequestersNotified` `true`.

A method on the `PersonalDataDestruction` plugin, `notifyExternalSystemsRequestProcessed`, is called by the class `PersonalDataDestructionNotifyExternalSystemsWorkQueue` to notify external systems when a personal

data destruction request is completed. The original RequestID is passed to the method, which does nothing by default. You are expected to implement this method to notify systems of interest. The RequestID is received when the destruction request is originally created through PersonalDataDestructionAPI.

See also

- “Personal data destruction plugin implementation classes” on page 233

RemoveOldContactDestructionRequest work queue

This work queue finds all PersonalDataDestructionRequest, PersonalDataContactDestructionRequest, and PersonalDataDestructionRequester objects that have the following values:

- RequestersNotified set to true
- PersonalDataContactDestructionRequest.DestructionDate plus the value of the ContactDestructionRequestAgeForPurgingResults configuration parameter is less than or equal to today's date
- Each found request that has AllRequestsFulfilled equal to true is removed.

Note: The class that implements this work queue is RemoveOldContactDestructionRequestWorkQueue.

PersonalDataDestructionController class

This class handles the complete lifecycle of the asynchronous destruction process after a destruction request has been made through the PersonalDataDestructionAPI web service. This class attempts to destroy the contact and notify the requesters that the contact has been destroyed.

The class provides the following methods relating to the lifecycle:

notifyRequesterDestructionRequestHasFinished(destructionRequester: PersonalDataDestructionRequester)

Takes a requester and notifies the external system that the request with related AddressBookUID and PublicID values was processed and finished.

destroyContact(contactDestructionRequest: PersonalDataContactDestructionRequest)

Called by PersonalDataContactDestructionWorkQueue, the class that implements the DestroyContactForPersonalData work queue, when attempting destruction. Uses the class that implements the PersonalDataDestroyer interface, called the Destroyer, to destroy the contact, and returns a ContactDestructionStatus. Verifies status and sets appropriate PersonalDataContactDestructionRequest and PersonalDataDestructionRequest attributes.

requeueContactRemovalRequestWithPublicID(publicID : String, bundle : Bundle)

Sets purge request status to ReRun after manual intervention, allowing the purge request to be reattempted for destruction.

See also

- “Data destruction web service” on page 219
- “Defining the Destroyer” on page 231

Data model configuration for data destruction

There are data model configurations that apply to the objects being destroyed. Some are general configurations, and some are specific to purging or obfuscation.

DestructionRootPinnable delegate

An entity that implements this delegate gets a `DoNotDestroy` flag that can be checked as part of the destruction process. An entity that is intended to be the root of an entity graph must implement the `DestructionRootPinnable` delegate if it is to be used in personal data destruction.

Root of entity graph for personal data destruction

The destruction process uses an entity domain graph to determine what to destroy. An object that implements the `DestructionRootPinnable` delegate and the `RootInfo` delegate is the root of an entity domain graph.

`ContactManager` has one root entity, `ABContact`.

Do Not Destroy flag

A Do Not Destroy flag is provided in the `DestructionRootPinnable` delegate. If an entity implements this delegate, instances of the entity have a `DoNotDestroy` Boolean field. The default value of this field is `false`.

Note: If this field is on a `Contact` entity or a subentity of `Contact`, it is maintained only in `ContactManager`. Even if the contact is linked, the field is not sent to `ContactManager`, nor is it updated from `ContactManager`.

In the base configuration of `ContactManager`, the `ABContact` entity implements the `DestructionRootPinnable` delegate and therefore has a `DoNotDestroy` Boolean field. The default value of the field is `false`, which permits destruction of the entity. If this field is set to `true`, the entity cannot be purged.

You can set this field for `ABContact` and `Contact` objects. Use the `markDoNotDestroy` method to set the field. For example:

```
ABContact.markDoNotDestroy(true)
```

Do Not Process flag

A Do Not Process flag is provided by the `Operable` delegate. If an entity implements this delegate, the entity then has a `DoNotProcess` Boolean field that can you use as needed in your configuration of `ContactManager`. This flag is not used in the base configuration.

Display keys for data destruction messages

There are a number of display keys defined for use by the personal data destruction code. You can see display keys for all languages supported by Guidewire in Guidewire Studio, such as the U.S. English display key properties. Navigate in the **Project** window to **configuration**→**config**→**Localizations**→**Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor. In this file, press `Ctrl+F`, enter the search string `PersonalData`, and then click the down or up arrow button to go to each search result in the file.

For example:

```
PersonalData.Error.ObfuscateNotImplemented = Obfuscation for '{0}' is not supported
```

Obfuscatable delegate

If you intend to use obfuscation with an entity, it must implement the `Obfuscatable` delegate. This delegate is necessary to support marking fields as personally identifiable information with the `PersonalData` tag.

Note: A `Delegate` is a reusable type that defines database columns, an interface, and a default implementation of that interface. A delegate permits an entity to implement an interface while delegating the implementation of that interface to another class, that of the delegate. Each delegate type provides additional columns on the affected tables.

The `Obfuscatable` delegate has one column, `ObfuscatedInternal`, which cannot be set in Gosu code. This limitation exists because after the `ObfuscatedInternal` column has been set to `true`, it must not be set to `false`.

Note: If the parent of an entity implements the `Obfuscatable` delegate, all child entities inherit that implementation.

The `Obfuscatable` delegate extends the interfaces `Obfuscator` and `ObfuscatablePublicMethods`. These interfaces provide the following methods:

markAsObfuscated

Marks the entity instance as having been obfuscated by setting `ObfuscateInternal` to true on the delegate.

isObfuscated

Returns true if `obfuscate` has been called and completed successfully. Otherwise returns false.

obfuscate

Obfuscates the columns that are marked for obfuscation on this object.

obfuscateSimple

Works the same as `obfuscate`.

Obfuscator interface

If you intend to use obfuscation with an entity, the entity must implement the `Obfuscator` interface. Each entity that implements the `Obfuscator` interface must also designate an implementation class, such as `UserContactObfuscator`.

Note: If the parent of an entity implements the `Obfuscator` interface, all child entities inherit that implementation, including the implementation class. You can override the parent implementation by declaring `implementsInterface` in the child entity.

For example, in the base configuration, `UserContact.etx` has the following definition:

```
<extension xmlns="http://guidewire.com/datamodel"
  entityName="UserContact">
  <implementsInterface
    iface="gw.api.obfuscation.Obfuscator"
    impl="gw.personaldata.obfuscation.UserContactDefaultObfuscator"/>
  <column-override
    name="EmployeeNumber">
    <tag
      name="PersonalData"
      value="ObfuscateDefault"/>
    </column-override>
  </extension>
```

In the base configuration, an entity can implement the `Obfuscatable` delegate but have an `Obfuscator` interface implementation of `UnsupportedObfuscator`. In this case, even if the entity is marked as obfuscatable, any attempt to obfuscate it results in an `UnsupportedOperationException`.

To be able to obfuscate an entity, you must either use an existing obfuscator or write a suitable obfuscator, such as one that extends `PersonalDataObfuscator`.

See also

- “Personal data obfuscation classes” on page 236
- “Implementing the `Obfuscator` interface in an entity” on page 235

Marking entity fields for obfuscation

An entity that implements the `Obfuscatable` delegate must also have fields tagged for obfuscation. For each field that contains personal data, add the tag `PersonalData`. The value of the tag determines the kind of obfuscation that will be applied to the field's contents. In the base configuration, two values are defined in the typelist `PersonalDataTagValue.tti`:

ObfuscateDefault

The field's contents are replaced with the default value or null if no default value is defined.

ObfuscateUnique

The field's contents are replaced as you define.

In the base configuration, all the entities that implement the `Obfuscator` interface have fields tagged `PersonalData`. Additionally, many of the fields tagged as `PersonalData` are in `.etx` files to enable you to modify them. For example, the `Name` field of `Contact` is defined in `Contact.etx` as follows:

```
<column-override
  name="Name">
  <tag
    name="PersonalData"
    value="ObfuscateUnique"/>
</column-override>
```

In addition, the following delegates have fields tagged `PersonalData`. An entity that implements one of these delegates does not have to implement `Obfuscator` and `Obfuscatable` unless you want the entity to support obfuscation.

- `AddressBookConvertible.etx`
- `GlobalAddress.etx`
- `GlobalAddress.Global.etx`
- `GlobalContactName.etx`
- `GlobalContactName.Global.etx`
- `GlobalPersonName.etx`
- `GlobalPersonName.Global.etx`

See also

- For a list of entities that implement `Obfuscator`, see “Implementing the `Obfuscator` interface in an entity” on page 235

ContactManager entity domain graph

`ContactManager` uses an `ABContact` entity domain graph to define the aggregate cluster of associated objects that it treats as a single unit for purposes of data destruction.

Overview of the ContactManager entity domain graph

The aggregate cluster of associated objects in an entity domain graph has a root and a boundary.

- The *root* is a single specific entity that the aggregate cluster contains. The root entity is the main entity in the graph. A root entity is application-specific and must implement `DestructionRootPinnable`. *Pinnable root* is another term for one of these root entities.

In the base configuration of `ContactManager`, the root entity is `ABContact`.

- The *boundary* defines what is inside the aggregate cluster of objects. It identifies all the entities that are part of the graph.

In `ContactManager`, the boundary defines the entities that relate to an `ABContact` object, such as `Address`, `ABContactSpecialistService`, `ABContactContact`, and so on.

The unit of work for the destruction process is a single instance of the domain graph, such as a single `ABContact` object and all its associated entities.

To enforce the boundaries of the domain graph, all objects participating in the destruction process must implement the `Extractable` delegate.

You cannot define a new entity domain graph for use in personal data destruction. However, you can configure an existing entity domain graph. For example, use the `archivingOwner` data model attribute and the `CrossDomainPublicID` data model tag.

At server startup, the entity domain graph is validated. If the domain graph fails validation, the action taken depends on whether or not personal data destruction is enabled.

- If the configuration parameter `PersonalDataDestructionEnabled` is set to `true`, a failed graph validation prevents the server from starting. In this case, you must make graph corrections. Without a proper graph, `ABContact` purge will not work.
- If the configuration parameter `PersonalDataDestructionEnabled` is set to `false`, a failed graph validation logs warning messages indicating that a graph is invalid and needs correction. However, the server does start up. If you do not use personal data destruction, you can safely ignore these warnings.

ContactManager ABContact entity domain graph

The root of the `ContactManager` entity domain graph is `ABContact`. The entity domain graph includes entities such as `Address`, `ABContactSpecialistService`, `ABContactContact`, and so on. In the base configuration, this entity graph is used for purging `ABContact` data. `ContactManager` does not support archiving.

`ContactManager` defines specifics of handling the `ABContact` entity domain graph itself in the Java class `ABDomainGraphSupport`. For example, this class has methods like `register`, `getDomainGraphName`, `getLinkFromRootInfoToRoot`, and so on. You are not likely to need to use any of the methods in this class.

The class `ABContactPurgeMethodsImpl` is provided for special handling of entities that are not directly part of the domain graph but need to be considered in a purge. For example, an entity has a foreign key pointing to an entity in the domain graph, but there is no reciprocal foreign key from the entity in the domain graph.

ABContact domain graph and related entities

There are several ways that an entity can be considered part of the `ABContact` domain graph. There are also entities that require special handling or are explicitly excluded from the graph.

- The entity is included in the purge graph because it implements `Extractable`.
See “Entities that implement `Extractable`” on page 228.
- The entity is included in the purge graph because it has a foreign key to `ABContact`.
See “Entities with foreign keys to an `ABContact` object” on page 229.
- The entity might qualify for purging, but requires special handling.
See “Entity requiring special purge handling” on page 229.
- The entity is a shared resource and is therefore excluded from the purge graph.
See “`ABContact` graph and entities that are shared resources” on page 229.

Entities that implement `Extractable`

The following entities are defined as part of the domain graph and are automatically purged along with `ABContact`:

- `PendingContactChange`
- `PendingContactCreate`
- `PendingContactUpdate`
- `ABContactCategoryScore`
- `ContactHistory`
- `TrackedChange`
- `ABContactTag`
- `ABContactSpecialistService`
- `ABContactDocumentLink`
- `EFTData`
- `ReviewSummary`
- `ReviewSummaryCategoryScore`
- `Address` – Addresses are connected to the domain graph in two ways, directly defined in `ABContact` as a `PrimaryAddress` or as part of `ABContactAddress`. Addresses are not a shared resource among `ABContact`

objects and are exclusive to the ABContact they are associated with, so they can be safely purged along with the ABContact.

- ABContactAddress
- ABContactContact – Acting as a specialized edge foreign key, this object references ABContact through the SrcABContactID and the RelABContactID fields. In the base configuration, these fields are cross domain links, which can be safely severed from the other contacts.
- MergeContactPair – Merge contact pairs are automatically purged along with ABContact, which will automatically severs relationships between that contact and any contacts that are duplicates. There is special handling for a contact that is marked as a retired MergeContactPair object.

MergeContactPair objects represent a sort of versioning system for contacts. In the system there can be a number of contacts that, while different objects, really represent the same thing. The system has the ability to denote one version of the contact as the *kept* version, the official representation of the contact. The other versions are demoted and retired, but the system is able to keep track of them. These obsolete versions are called *retired* versions, and you can think of them as replaced by the kept version. MergeContactPair entities are used by ContactManager to help represent this kept and retired versioning.

There are special conditions, though, that apply to the ABContact object itself when a MergeContactPair exists:

- If the contact being purged is the kept contact, then the purge destroys not just it, but also any retired versions of that contact.
- If the contact being purged is a retired contact, attempting to purge that contact causes an exception to be thrown. The exception informs the user that deleting the contact is prohibited, and that deleting the contact requires explicitly purging the kept version of that contact.

Entities with foreign keys to an ABContact object

The following entities have foreign keys that point to an ABContact object, but there is no foreign key from the ABContact object to the entity. These entities do not implement Extractable and are purged explicitly by ContactManager along with ABContact.

- ABContactScoringWorkItem – Deleted during the purge by ContactManager.
- DuplicateContactWorkItem – Any that reference the contact are purged by ContactManager. Even though the entity references ABContact by using a non-nullable foreign key, deleting the entity is not flagged as problematic by the graph validator.
- MessageHistory – Deleted during the purge by ContactManager.
- Message – Deleted during the purge by ContactManager.

Entity requiring special purge handling

The DuplicateContactPair entity requires special handling. The ABContact entity cannot be purged if the contact continues to be referenced by any DuplicateContactPair.

ABContact is referenced by two non-nullable foreign keys, ContactID and DuplicateContactID. These two foreign keys are marked as cross domain, which makes DuplicateContactPair severable.

Purging will fail and throw an exception if an ABContact is referenced by any DuplicateContactPair entity, either as the contact or as the duplicate contact. A user of ContactManager must use the **Merge Duplicates** screens to resolve duplicates first. Only when there are no references will purging be allowed.

ABContact graph and entities that are shared resources

Resources that are shared can also be system or administration entities. The distinction does not matter. In terms of the domain graph, these objects are shared resources and are not part of the graph. These objects are not purged and remain in the system under all conditions. The ABContact object has the following shared resources:

- DuplicateContactBatchRun – These objects are indirectly part of the ABContact graph.
- SpecialistService
- SpecialistServiceParent
- Document – A document can be purged only if it is not shared by another ABContact object. Document metadata entities can be shared by one or more contacts. If only a single contact is related to a document, then the

Document entity is purged along with the contact. Purging the document in this case is required by a database consistency check in ContactManager.

Table ab_purgedrootinfo

Whenever ContactManager purges an ABContact, it deletes the ABContact domain graph instance for that contact. To track the contacts it purged, ContactManager creates a PurgedRootInfo entity instance at the same time that it purges a contact. This entity records the publicID of the purged contact. ContactManager then stores each PurgedRootInfo instance as a row in the table ab_purgedrootinfo.

Note: ContactManager does not automatically delete PurgedRootInfo rows from the table. If you have a high purge volume, Guidewire recommends that you create a batch process to delete old, unwanted PurgedRootInfo instances from the ab_purgedrootinfo table.

Extensions of ABContact and other entities and the domain graph

If you have extended the ABContact data model, personal data purge is able to handle your extensions as long as you address domain graph design guidelines.

You can also create a variety of relationships between entities through foreign keys. Again, to support personal data purge, follow the guidelines for managing entities through the domain graph.

Following are some general guidelines:

- An entity is referenced by the ABContact or is part of the direct contact graph. The entity must implement `Extractable`, which makes the entity part of the contact graph. The entity will automatically be destroyed along with the rest of the contact graph when the personal data purge initiated. This guideline also applies to entities that are used as array objects.
- An ABContact object or other entity in the contact graph has a foreign key pointing to an entity, but the entity being pointed to has no reciprocal foreign key. The entity being pointed to is not part of the graph and is not deleted as part of personal data purge. Special handling is optional. You can add special handling to perform cleanup during the personal data purge.
- An entity has a foreign key that points to an ABContact entity or another entity that is part of the domain graph. However, there is no reciprocal foreign key from the domain graph entity. In this case, special handling is required because deleting the domain graph object leaves nothing for the original entity's foreign key to reference.

You can code your special handling in the `ABContactPurgeMethodsImpl` class.

ContactManager Data Protection Officer

A Data Protection Officer is expected to be available to handle problems with data destruction. Therefore, in the base configuration, there is a Data Protection Officer role to which users can be assigned. The code name for this role is `data_protection_officer`.

Data Protection Officer permissions

In the base configuration of ContactManager, the Data Protection Officer role has the following permissions relating to personal data destruction:

- `requestcontactdestruction`
- `editobfuscatedusercontact`

This role also has additional permissions to enable the user to work with users and groups. These permissions include `groupcreate`, `groupdelete`, `groupedit`, `usereditattrs`, `usereditlang`, `useredit`, `usergranroles`, `userviewall`, `grouptreereview`, `grouppreview`, and `userview`.

A user named DataProtection Officer with login `dpofficer`, which has the Data Protection Officer role, is available in the sample data.

In the base configuration, ContactManager screens prevent a user from editing obfuscated user contacts if the user does not have permissions to do so. In addition, ContactManager prevents a user without the correct permissions

from adding obfuscated user contacts to or removing them from groups and roles. You can create additional permissions and configure ContactManager to further limit editing of obfuscated objects.

Notifying the Data Protection Officer

The `PersonalDataDestruction` plugin interface provides a method that enables notification of the Data Protection Officer, `notifyDataProtectionOfficer`.

For example, in the base configuration, the class that implements the `PersonalDataDestruction` plugin interface, `ABPersonalDataDestructionSafePlugin`, overrides the `notifyDataProtectionOfficer` method. In this class, the `notifyDataProtectionOfficer` method logs messages to the system console if a destruction request fails.

Note: The class `ABPersonalDataDestructionSamplePlugin` has the same implementation for the `notifyDataProtectionOfficer` method.

Data destruction purge configuration

Purging is the process of completely removing `ABContact` data from the ContactManager database. There can be multiple objects associated with a contact that are also removed as they are detected by traversing the entity domain graph.

Note: ContactManager is expected to be the last Guidewire application web service called by the external system to process a personal data destruction request. When ContactManager receives a contact destruction request from the web service, it makes calls to the core application implementation of `ContactAPI.isContactDeletable` to determine if the contact can be destroyed. The contact will be destroyed in ContactManager only if all the applications return a positive response saying the contact can be destroyed.

See also

- “ContactManager entity domain graph” on page 227
- “Data obfuscation in ContactManager” on page 234

Defining the Destroyer

The `ABPersonalDataDestroyer` class implements the `PersonalDataDestroyer` interface and provides methods that determine how a destruction request is carried out. The class provides methods that are called by the `PersonalDataDestructionAPI` web service and its work queues.

A personal data destruction request to delete a contact by `AddressBookUID` corresponds in ContactManager to one `ABContact` with a single `LinkID` value. The method `translateABUIDToPublicIDs` finds the `PublicID` value of the contact and returns it to the web service, which creates a `PersonalDataContactDestructionRequest`. This personal data destruction request enables the work queue to make a `destroyContact` method call to delete the contact by `PublicID` when the work item is processed.

If a personal data destruction request specifies deleting a contact by `PublicID`, ContactManager can directly use the `PublicID` to delete the `ABContact` object.

The following two methods are the primary Destroyer methods used by the web service.

`translateABUIDToPublicIDs`

This public method overrides the `PersonalDataDestroyer` interface method `translateABUIDtoPublicIDs`. It finds the `ABContact` with a `LinkID` that is the same as the specified `AddressBookUID` and returns the contact's `PublicID` value.

The method is called by the web service `PersonalDataDestructionAPI`. The web service uses the method to determine if an `AddressBookUID` exists and to get the `PublicID` if the original destruction request was specified by `AddressBookUID`.

destroyContact by Destruction Request

This public method overrides the `PersonalDataDestroyer` interface method `destroyContact`. It is the main entry point for destroying contacts.

Note: The `PersonalDataContactDestructionWorkQueue` class calls this method when it processes a work item. The class calls `PersonalDataDestructionController.destroyContact(contactPurgeRequest)`.

This method takes a `PersonalDataDestructionRequest` and finds the corresponding `ABContact` by `PublicID`. Once the `ABContact` is found, a call to the `PersonalDataDestruction` plugin is made to determine whether the contact can be purged.

- If the result of the plugin call is `MUST_NOT_DESTROY`, the request is not processed for destruction. The reason is logged in the `DATA_DESTRUCTION_REQUEST` logger and the Destroyer returns the status `ContactDestructionStatus.TC_NOTDESTROYED`.
- If the result of the plugin call is `MAY_DESTROY` or `MUST_DESTROY`, the request is processed for destruction and `destroyContact(contact)` is invoked.
 - If the purge is successful the Destroyer must return the status `ContactDestructionStatus.TC_COMPLETED`.
 - If there is a purge error, the Destroyer must return the status `ContactDestructionStatus.TC_MANUALINTERVENTIONREQUIRED` and log the error or exception in the `DATA_DESTRUCTION_REQUEST` logger.

Note: The `notifyDataProtectionOfficer` method uses the error log level if there is a purge error. Otherwise it just uses the info log level.

It is possible that the contact was previously purged and cannot be found. For example, a previous `PersonalDataDestructionRequest` resulted in the `ABContact` being purged. In that case, the method returns `ContactDestructionStatus.TC_COMPLETED`.

See also

- “Data destruction web service” on page 219

PersonalDataPurge event

When a personal data purge of certain objects is committed, `ContactManager` generates a `PersonalDataPurge` event that you can respond to in an Event Fired rule to send a message. For example, you might want to send a message to a downstream system.

`ContactManager` generates a `PersonalDataPurge` event when an `ABContact` purge is committed as part of a personal data purge. In the base configuration, there is no rule that responds to this event, but you can create an EventFired rule in the EventMessage rule set category that sends a message.

Your rule could take the following form in the Guidewire Studio Rules editor:

USES:

CONDITION (messageContext : entity.MessageContext):

```
return messageContext.EventName == "PersonalDataPurge"
```

ACTION (messageContext : entity.MessageContext, actions : gw.Rules.Action):

```
var claimInfo = messageContext.Root as ABContact
messageContext.createPersonalDataPurgeMessage(
  "ABContact with public ID ${contact.PublicID} has been successfully purged")
```

See also

- “ContactManager messaging events” on page 269
- For general information on messaging and message events, see the *Rules Guide*

Specifying which objects in the graph can be destroyed

The `PersonalDataDestruction` plugin interface provides basic methods that define which objects can be destroyed and how to contact the Data Protection Officer. You define a plugin implementation class to define this behavior and register it in the `PersonalDataDestruction.gwp` plugin registry.

PersonalDataDestruction plugin interface

This interface provides the following methods for implementation by a `ContactManager` plugin implementation class:

```
PersonalDataDisposition shouldDestroyRoot(
    Pinnable root,
    Collection<Pinnable> descendants, Pinnable origin);
PersonalDataDisposition shouldDestroyUser(UserContact userContact);
void notifyDataProtectionOfficer(
    Pinnable root, String title, String message, Date dateOfError);
void notifyExternalSystemsContactHasBeenPurged(
    String AddressBookUID, String requestor, String requestID);
PersonalDataPurgeContext createContext(PersonalDataPurgeContext context);
void prepareForPurge(PersonalDataPurgeContext context);
void postPurge(PersonalDataPurgeContext context);
PersonalDataDestroyer getDestroyer();
```

In the base configuration, `ContactManager` registers the `ABPersonalDataDestructionSafePlugin` class as the default class that implements `PersonalDataDestruction`, which prevents destruction of any of the pinnable root entities. A more realistic starting point for your purge definition is the `ABPersonalDataDestructionSamplePlugin` class.

See also

- “Personal data destruction plugin implementation classes” on page 233
- “ContactManager entity domain graph” on page 227

Personal data destruction plugin implementation classes

In the base configuration of `ContactManager`, the `ABPersonalDataDestructionSafePlugin` class is registered as the class that implements the `PersonalDataDestruction` plugin interface. This class provides default handling for destruction of pinnable root entities in the base configuration. It prevents destruction of any personal data.

`ABPersonalDataDestructionSamplePlugin` is the class you can use as an example when you implement your own personal data destruction class to define both `getDestroyer` and how specific pinnable roots are handled. You must then register your implementation class with the plugin registry `PersonalDataDestruction.gwp`.

These two classes define methods that control destruction of pinnable root entities by returning one of the values defined in the enum `PersonalDataDisposition`:

- `MUST_NOT_DESTROY` – The object must not be destroyed. If this value is in conflict with a `MUST_DESTROY` value in the domain graph, the Data Protection Officer must get involved.
- `MUST_DESTROY` – The object must be destroyed.
- `MAY_DESTROY` – The object can be destroyed.

ABPersonalDataDestructionSafePlugin

In the base configuration, `ABPersonalDataDestructionSafePlugin` calls `getDestroyer` to obtain the destroyer defined in `ABPersonalDataDestroyer`. Additionally, this class prevents data destruction by returning `MUST_NOT_DESTROY` for all calls to destroy pinnable root entities. For example:

```
override function shouldDestroyRoot(
    root: DestructionRootPinnable,
    descendants: Collection<DestructionRootPinnable>,
    origin: DestructionRootPinnable): PersonalDataDisposition
{
    notifyDataProtectionOfficer(
```

```

        root, "Safe plugin implementation always returns MUST_NOT_DESTROY")
    return MUST_NOT_DESTROY
}
override function shouldDestroyUser(
    userContact: UserContact): PersonalDataDisposition
{
    notifyDataProtectionOfficer(
        userContact, "Safe plugin implementation always returns MUST_NOT_DESTROY")
    return MUST_NOT_DESTROY
}
private function notifyDataProtectionOfficer(
    contact : DestructionRootPinnable, message : String)
{
    notifyDataProtectionOfficer(contact, null, message, null)
}
override function notifyDataProtectionOfficer(
    root: DestructionRootPinnable, title: String, message: String, dateOfError: Date)
{
    ABPersonalDataLogUtil.logInfoNotDestroyed(root, message)
}

```

ABPersonalDataDestructionSamplePlugin

You can use the class `ABPersonalDataDestructionSamplePlugin` as a guide for writing your own personal data destruction code. This class can return values other than `MUST_NOT_DESTROY` for a pinnable root entity.

The class returns `MUST_NOT_DESTROY` in the following circumstances:

- The contact is an `ABContact` with `DoNotDestroy` set to `true`.
- The subtype of the contact is `ABCompany` or `ABPlace`.
- Core applications were checked for permission to destroy the contact, and at least one application did not permit the destruction.

The class returns `MUST_DESTROY` if the contact:

- Is an `ABContact` for which `DoNotDestroy` is `false`, the subtype is not `ABCompany` or `ABPlace`, and all core applications permit destruction of the contact.
- Is a `UserContact`.

See also

- “Do Not Destroy flag” on page 225
- “DestructionRootPinnable delegate” on page 225
- “ContactManager entity domain graph” on page 227

Data obfuscation in ContactManager

In general, personal data destruction in `ContactManager` is done through removal of database records. However, the entities associated with an employee who works in your installation of `ContactManager` are not conducive to deletion because of the way data creation and changes are recorded. In particular, objects in the database are connected to the user that created them, and, in many cases, the user that last modified them.

Because an employee is likely to create or modify hundreds of thousands of objects, it would be computationally expensive to locate all those objects in the database. It would also be expensive to change those references to something else. It is not necessary to destroy the relationship between all the work that the employee performed and the fact that it was performed by a specific employee. If the employee's personally identifiable data is destroyed, the set of objects associated with the employee can remain and not violate the need to destroy personal data.

Therefore, in the base configuration, `ContactManager` obfuscates data related to `UserContact` objects.

Obfuscated objects

Each object can detect if it has been obfuscated or not through its `Obfuscated` flag. The system has no special handling for objects that have gone through obfuscation. Obfuscated objects act like any other active object in the system regarding search results, batch processes, and so on. You can implement additional functionality to filter

obfuscated beans, according to ContactManager configuration capabilities. In your obfuscation implementation, you must take into account how your custom obfuscation might affect existing processes in ContactManager.

Preupdate rules

Data obfuscation works the same as a normal entity editing, so changes made during obfuscation will trigger preupdate rules for entity types that have rules registered.

Implementing the Obfuscatable delegate in an entity

To be obfuscatable, an entity must implement the `Obfuscatable` delegate. If the entity implements `DestructionRootPinnable`, the `DoNotDestroy` field is set to `false` by default, which enables the entity to be obfuscated.

If an entity implements this delegate, the fields to be obfuscated must be tagged `PersonalData`.

Note: Tagging is not supported for array references, nor does obfuscation cascade automatically through foreign keys. Arrays and cascading through foreign keys must be handled in Gosu code in the `Obfuscator` implementation class.

See also

- “Marking entity fields for obfuscation” on page 226
- “Implementing the Obfuscator interface in an entity” on page 235
- “Obfuscatable delegate” on page 225
- “Do Not Destroy flag” on page 225

Implementing the Obfuscator interface in an entity

To be obfuscatable, an entity must implement the `Obfuscator` interface and specify an implementation class other than `UnsupportedObfuscator`. For example, use a class that extends `DefaultPersonalDataObfuscator`.

The entities that have `Obfuscator` implementations that support obfuscation are:

- `Credential` – `CredentialDefaultObfuscator`
Has fields and typekeys marked `PersonalData`.
- `OfficialID` – `DefaultPersonalDataObfuscator`
Has fields and typekeys marked `PersonalData`.
- `User` – `UserDefaultObfuscator`
Has fields and typekeys marked `PersonalData`.
- `UserContact` – `UserContactDefaultObfuscator`
This entity is a subtype of `Person`, which is a subtype of `Contact`. It inherits the `Contact` implementation of the `Obfuscatable` delegate and overrides the `Contact` implementation of the `Obfuscator` interface. In the base configuration, the `EmployeeNumber` field is marked `PersonalData`.

The following entities implement the `Obfuscatable` delegate, and in the base configuration their `Obfuscator` interface implementation is `UnsupportedObfuscator`:

- `Address`
- `Contact` – Has fields and typekeys that are marked `PersonalData`.
- `ContactCategoryScore`
- `Contact.Global`
- `Person` – Inherits obfuscation settings from `Contact`. Has fields and typekeys that are marked `PersonalData`.

See also

- “Obfuscator interface” on page 226

Personal data obfuscation classes

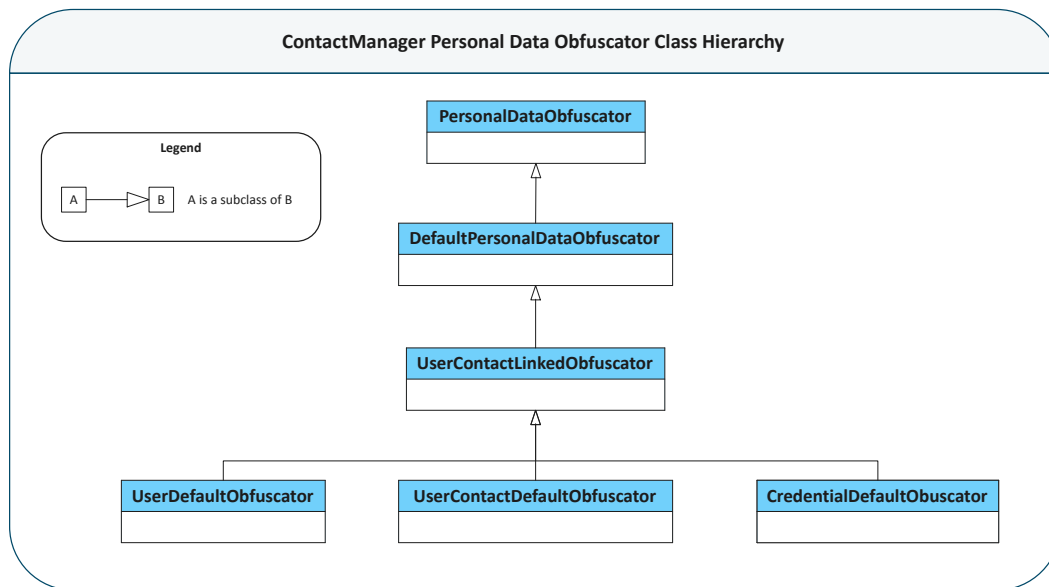
The personal data obfuscation classes, all of which ultimately inherit from `PersonalDataObfuscator`, define obfuscation for specific entities. The specific class that an entity uses is defined in its `implementsEntity` element for `gw.api.obfuscation.Obfuscator`.

See also

- “Implementing the Obfuscator interface in an entity” on page 235
- “Obfuscator interface” on page 226
- “Marking entity fields for obfuscation” on page 226

Personal data obfuscation class hierarchy

ContactManager provides the following class hierarchy for personal data obfuscation:



`DefaultPersonalDataObfuscator` and its subclasses define base configuration behavior to enable obfuscating `User`, `UserContact`, and `Credential` and related objects. For example:

- `DefaultPersonalDataObfuscator` overrides the method `getObfuscatedValueForPersonalDataField` and defines default obfuscation behavior for fields that are tagged `ObfuscateUnique`. The method handles `ObfuscateUnique` by calling `PersonalDataObfuscatorUtil` to get an MD5 hash value for the field.
- `UserContactLinkedObfuscator` overrides the `shouldObfuscate` method. That method calls the `shouldDestroyUser` method defined in the `PersonalDataDestruction` plugin to get the setting for `UserContact`. For example, in the base configuration the setting is `MUST_NOT_DESTROY`.
- `UserDefaultObfuscator` overrides a `beforeObfuscate` method. That method throws an exception if the user’s credential is active because that means the user is still active. If the user is not active, the method calls `user.Credential.obfuscate` and `user.Contact.obfuscate` and removes the arrays of join entities `AttributeUser` and `UserRegion`.
- `UserContactDefaultObfuscator` attempts to obfuscate or remove any contact addresses, official IDs, category scores, and tags associated with the `UserContact` that can be destroyed.
- `CredentialDefaultObfuscator` overrides the `beforeObfuscate` method, which stops the obfuscation if the credential is active.

PersonalDataObfuscator

`PersonalDataObfuscator` is the parent class for the obfuscator classes. It is a general class that obfuscates the fields for any entity. You extend this class or one of its subclasses when implementing obfuscation for entities that do not have obfuscator classes defined.

`PersonalDataObfuscator` handles setting the obfuscation for marked fields. If you have tagged all the entity fields `PersonalData` with value `ObfuscateDefault`, and there is no need to do any special preprocessing of the fields, you can use this class.

The following methods are provided:

- `isObfuscated` – Checks the field `ObfuscatedInternal` and returns `true` or `false` depending on the value.
- `obfuscate` – Finds all the columns that are marked for obfuscation on this object.
 - The method sets the field value with the `obfuscatedValue`.
 - If the column is marked `PersonalData` with a value of `ObfuscateDefault`, the method sets the value to either `null` or the default value.
 - If the column is marked `PersonalData` with a value of something other than `ObfuscateDefault`, the method calls `getObfuscatedValueForPersonalDataField`. In the base configuration, `getObfuscatedValueForPersonalDataField` is defined in `DefaultPersonalDataObfuscator` and uses the tag value. You can override the definition of this method to change how it obfuscates the field.
 - At the end, the method sets the `ObfuscateInternal` column to `true` by using `ObfuscatablePublicMethod.setObfuscated`.
- `obfuscateSimple` – Works the same as `obfuscate`.

If you want to do preprocessing before obfuscation, you can extend `PersonalDataObfuscator` or a subclass of this class and override the `beforeObfuscate` method. If you want to change the value of the personal data field before obfuscation, you can override `getObfuscatedValueForPersonalDataField`.

See also

- “Marking entity fields for obfuscation” on page 226

UnsupportedObfuscator

This Java class provides a default implementation for any entity that implements the interface `Obfuscator` and declares `UnsupportedObfuscator` as the implementation. When `obfuscate` is called, this class throws `unsupportedOperationException` for the field.

PersonalDataObfuscatorUtil

This Gosu class is in the same package as the personal data obfuscation classes, `gw.personaldata.obfuscation`. The class implements the method `computeMD5Padding`. If a personal data field has a `PersonalData` tag with value `ObfuscateUnique`, this method is called to obfuscate the field.

The method computes an MD5 String based on the type of entity and the `PublicID`, and then returns that string so the field can be obfuscated with that value.

For example, `DefaultPersonalDataObfuscator` calls this method in its `getObfuscatedValueForPersonalDataField` method when the field's `PersonalData` tag has the value `PersonalDataTagValue.TC_OBFUSCATEUNIQUE.Code`.

See also

- “Marking entity fields for obfuscation” on page 226

Working directly in ContactManager

You can work directly with contact data in ContactManager by using screens that manage contact data and ContactManager functionality.

[See also](#)

Logging in to ContactManager

You log in to ContactManager by running the application and logging in with your user name and password.

ContactManager login requirements

Logging in to ContactManager requires the following:

- **A web browser** – Guidewire supports Firefox, Google Chrome, and Microsoft Internet Explorer.
- **The URL (web address) for connecting to ContactManager**
 - In the base configuration, the URL is `http://localhost:8280/ab`
 - You can set up a **Favorite** link to the URL or create a shortcut on your computer desktop that starts your web browser with that URL.
- **A user name and password** – You must have one or more roles assigned to your user name by a system administrator. Roles determine the screens you can access and what you can do in ContactManager.

Because ContactManager generates screens dynamically:

- You cannot create **Favorites** to screens other than the login screen.
- The **Back** button of the browser is not supported.

[See also](#)

- For details on installation and the web address, see “Installing ContactManager” on page 55.

Log in to ContactManager

Procedure

1. Launch ContactManager by running a web browser and using the appropriate web address, such as:

```
http://localhost:8280/ab
```

2. Enter your **User Name** and **Password** on the login screen.

Result

If your login is successful, ContactManager shows your startup view, or landing page. In the default configuration, ContactManager initially opens the **Search** screen on the **Contacts** tab.

Change your password in ContactManager


Before you begin

Log in as described at “Log in to ContactManager” on page 239.

About this task


After you log in to ContactManager, you can change your password.

Procedure

1. On the Options menu , click **Preferences**.
The **Preferences** worksheet opens below the main work area.
2. Enter your old password.
3. Enter the new password.
4. Enter the new password again in the **Confirm New Password** field.
5. Click **Update**.

Change your user preferences in ContactManager

Procedure

1. Log in to ContactManager.
2. On the Options menu , click **Preferences**.
The **Preferences** worksheet opens below the main work area.
3. Change your preferences, such as your password, regional formats, default country, and default phone region.

Next steps

See “Preferences worksheet” on page 240.

Preferences worksheet


In the **Preferences** worksheet, you can change your password, set your regional formats, your default country, and your default phone region. You set the last three if you want your personal settings to be different from the ones set for all users of ContactManager.

- **Password** – Reset your password. See “Change your password in ContactManager” on page 240.
- **Regional Formats** – Set the regional formats that ContactManager uses to enter and display dates, times, numbers, monetary amounts, and names.
- **Default Country** – Determine the settings for names and addresses.
- **Default Phone Region** – Determine how phone number entries are handled, especially the country code setting.

Selecting international settings in ContactManager

In ContactManager, each user can set the following:

- The language that ContactManager uses to display labels and drop-down menu choices
- The regional formats that ContactManager uses to enter and display dates, times, numbers, monetary amounts, and names.

You set your personal preferences for display language and for regional formats by using the Options menu  at the top, right-hand side of the ContactManager screen. On that menu, click **International**, and then select one of the following:

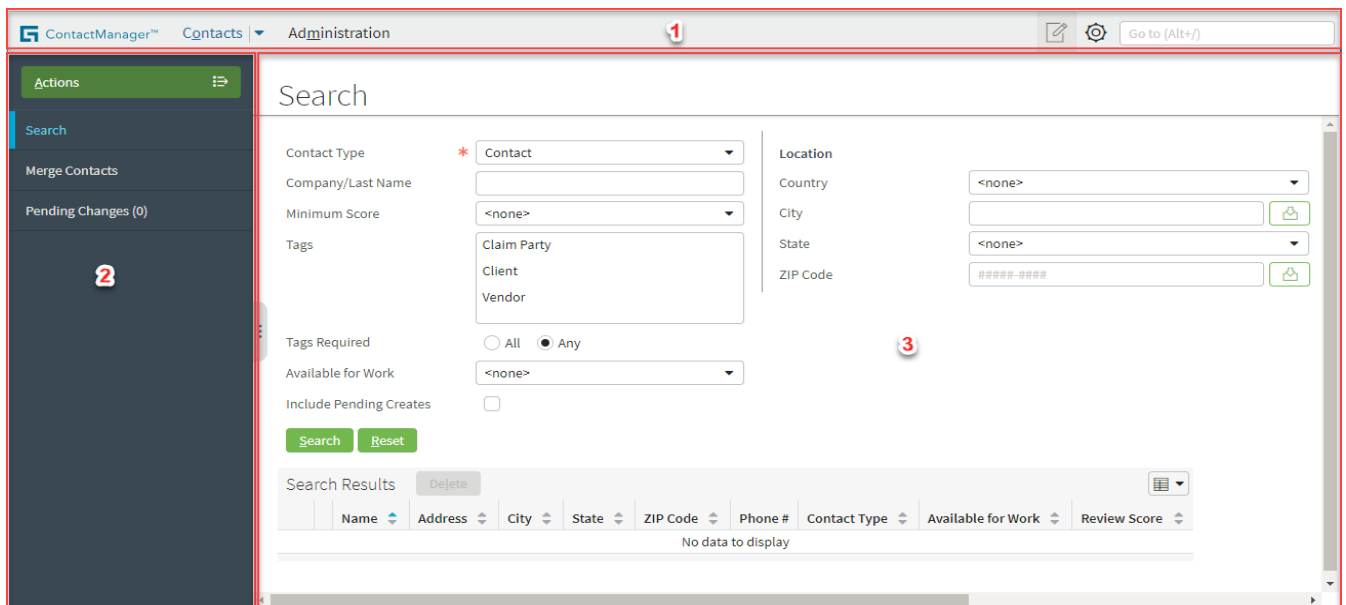
- **Language**
- **Regional Formats**

See also


- “International settings in ContactManager” on page 243


ContactManager user interface

Like the Guidewire core applications, ContactManager has tabs at the top of the screen and an **Actions** button and a menu on the left for navigation.



The ContactManager main user interface contains the following areas:

Area	Description
1	<p>The Tab Bar contains:</p> <ul style="list-style-type: none"> • Tabs, which group major areas of work for you, providing main screens, actions on the Actions menu, and Sidebar menu links. The tabs you see depend on your user permissions. All users can see the Contacts tab. • Unsaved Work button . This button is active if you have work you have not saved. You can click the button and go to any screen in which you have not saved your changes. This feature is useful if you must navigate away from a screen in which you are making changes, but you intend to return to it later. After you complete and save your work, ContactManager removes that item from the Unsaved Work drop-down menu.

Area	Description
	<ul style="list-style-type: none"> The Options menu . The selections available on this menu depend on where you are in the ContactManager user interface. Typical menu choices are International, Help, About, Preferences, Clear Layout Preferences, and Log Out Username. QuickJump box. The text box that displays Go to (Alt+I). In the base configuration of ContactManager, this box enables you to run batch processes with the command <code>RunBatchProcess batchProcess</code>. You can configure it.
2	The Sidebar contains menu links and the Actions menu. Use the Sidebar menu links to navigate to screens where you can do your work. The items in the Sidebar are contextual and change depending on tab and menu selections.
3	The Screen Area shows most of your business information and is where you interact with that information.

See also

- For a description of the **International** menu and its submenu selections, **Regional Settings** and **Language**, see “International settings in ContactManager” on page 243.
- For information on using and configuring **QuickJump**, see the *Application Guide*.

Contacts tab

The **Contacts** tab enables you to:

- Click **Actions** to create a new person, company, or place to store in ContactManager, if you have permissions to do so.
- Click **Search** to find contacts by using search criteria, such as contact subtype and location.
- Edit and delete any contact entities for which you have permissions.
- Click **Merge Contacts** to merge duplicate contacts that have been discovered by Duplicate Contacts Finder batch processing, if you have permissions to do so.
- Click **Pending Changes** to review changes to contacts that were made by core application users who did not have permissions to make the changes. You need the correct permissions to do so.

See also

- “Detecting and merging duplicate contacts” on page 247
- “Review pending changes to contacts” on page 253

Administration tab

On the ContactManager **Administration** tab, you can manage roles, permissions, login name and password, and group membership for your ContactManager users. Additionally, you can manage message queues, export and import data, and work with script parameters.

Note: To see this tab, you must be logged in to ContactManager as an administrator.

- The **Actions** button on the left enables you to create new ContactManager users and groups.
- Also on the left in the Sidebar is a hierarchical list of all your groups and users.

Additionally, you can choose the following items from the Sidebar:

- Users and Security** – Groups the following administrative tasks:
 - Users** – Search for users by role, location, username, and first and last name.
 - Groups** – Search for groups by name and type.
 - Roles** – Add and delete roles, add permissions to and remove permissions from roles, and assign roles to ContactManager users.
 - Regions** – Add, delete, and edit the current regions for ContactManager contacts. *Regions* are geographical areas you can use to define groups’ areas of responsibility. A region can contain one or more states, ZIP codes,

counties, or other address elements, such as Canadian provinces. You can assign users and groups to cover one or more regions, and then define business rules to provide location-based assignment.

- **Monitoring** – Groups the following administrative tasks:
 - **Message Queues** – Suspend and resume event messages sent by ContactManager and restart the messaging engine. One use of the **Message Queues** screen is to restart message queues that have been suspended. For example, the message queue for a core application might have been suspended because the core application was not running when ContactManager tried to send contact data.
- **Utilities** – Groups the following administrative tasks:
 - **Import Data** – Import certain types of data through the ContactManager interface.
 - **Export Data** – Export certain types of data through the ContactManager interface.
 - **Script Parameters** – View existing ContactManager script parameters. To create new script parameters, you must use ContactManager Studio.
 - **Data Change** – Enables you to push data changes to the production server. Use this feature sparingly and only to update mission-critical data on running production systems..

See also

- For information on roles and security, see “Securing access to contact information” on page 121.
- “ContactManager messaging events” on page 269
- “Troubleshooting the ClaimCenter connection with ContactManager” on page 65
- “Troubleshooting the PolicyCenter connection with ContactManager” on page 72
- “Troubleshooting the BillingCenter connection with ContactManager” on page 78
- For information on data change, see .
- For information on script parameters, see the *Configuration Guide*

Internal tools and server tools tabs

You can log in to ContactManager as the super user. You can then press Alt+Shift+T, click the **Internal Tools** tab, and load sample data.

Alternatively, you can click the **Server Tools** tab and perform administrative tasks, such as starting batch jobs manually.


See also

- “Load sample data for ContactManager” on page 56
- “Start work queues and batch processes” on page 255
- “Working directly in ContactManager” on page 239
- For more information on the Internal Tools and Server Tools tabs, see the *System Administration Guide*

International settings in ContactManager

You can set the following international settings:

- The language in which ContactManager displays labels and drop-down menu choices
- The regional formats that ContactManager uses to enter and display dates, times, numbers, monetary amounts, and names

You set your personal preferences for display language and for regional formats by using the Options menu  at the top, right-hand side of the ContactManager screen. On that menu, click **International**, and then select one of the following:

- **Language**
- **Regional Formats**

To set international settings in the application, you must configure ContactManager with more than one region. Your configuration of regions and languages determines what you see on this menu, as follows:

- ContactManager hides the **Language** submenu if only one language is installed.
- ContactManager hides the **Regional Formats** submenu if only one region is configured.
- ContactManager hides the **International** menu option entirely if a single language is installed and ContactManager is configured for a single locale.

ContactManager indicates the current selections for **Language** and **Regional Formats** by placing a check mark to the left of the selected options and displaying them in gray.

Options for language

In the base configuration, Guidewire has a single display language, English. To view another language in ContactManager, you must install a language pack and configure ContactManager for that language. If your installation has more than one language, you can select among them from the **Language** submenu. The `LanguageType` typelist defines the set of language choices that display on the menu.

If you do not select a display language from the **Language** submenu, ContactManager uses the primary display language. The configuration parameter `DefaultApplicationLanguage` specifies this language. In the base configuration, the primary display language is U.S. English.

Options for regional formats

If your installation contains more than one configured region, you can select a regional format for that region from the **Regional Formats** submenu. At the time you configure a region, you define regional formats for it.

Regional formats specify the visual layout of the following kinds of data:

- Date
- Time
- Number
- Monetary amounts
- Names of people and companies

The `LocaleType` typelist defines the names of regional formats that you can select from the **Regional Formats** submenu. The base configuration defines the following locale types:

- | | |
|-----------------------|---------------------------|
| • Australia (English) | • Germany (German) |
| • Canada (English) | • Great Britain (English) |
| • Canada (French) | • Japan (Japanese) |
| • France (French) | • United States (English) |

If you do not select a regional format from the **Regional Formats** menu, ContactManager uses the regional formats of the default region. The configuration parameter `DefaultApplicationLocale` specifies the default region. In the base configuration, the default region is `en_US`, United States (English). If you select your preference for region from the **Regional Formats** menu, you can later use the default region again only by selecting it from the **Regional Formats** menu.

See also

- For an introduction to globalization and more information on working with regional formats and languages, see the *Globalization Guide*

Changing the screen layout

You can adjust some aspects of the screen layout to fit your preferences. You can adjust list views, change the sidebar width, and manage layout preferences. The instructions for doing so in ContactManager are the same as those for the core applications.

See also

- For more information on changing the screen layout, see the *Application Guide*

Importing and exporting administrative data

While much administrative data is entered directly into ContactManager, there are times when it is useful to transfer this information in bulk.

The **Export Data** and **Import Data** screens available on the **Administration** tab provide a convenient way of moving administrative data and role definitions to and from XML files.

Additionally, you can export the security dictionary as either HTML or XML.

You can also import or export other types of data, in either XML or CSV format, by using an API. Also, you can use a command on the command line to import files in either format, but not to export them.

Method	Import	Export	File Formats
user interface	admin.xml	admin.xml, roles.xml, securitydictionarynumber.zip	XML, XML or HTML in compressed ZIP
command line	any	no	XML, CSV
API	any	any	XML, CSV

See also

- “Administration tab” on page 242

Import administrative data

About this task

You can import administrative data in the **Administration** tab. For more information, see the *System Administration Guide*.

Procedure

1. Click the **Administration** tab and then navigate to **Utilities**→**Import Data** to select a file to import.
2. You can click the **Browse** button to find the file.
For example, you have created a file of administrative data called `admin.xml`. The file must be either in XML or compressed XML format, with an XSD compatible with the XML files you can import. If you want to import administrative data, you can import any subset of this type of data, such as users or regions.
3. Click **Next** and follow the prompts to resolve differences between the data in the imported file and data already in the database.
Data not yet in the database is imported without question. If the imported data differs from what is already in the database, these prompts enable you either to accept the imported data or to keep what is in the database.
4. Click **Finish** to complete the import.

Exporting data in the administration tab

To export administrative data, click the **Administration** tab and then navigate to **Utilities**→**Export Data** to see the following categories. Each category has one or more types of data you can export. Each file contains all the data of a certain type in your installation. These export categories are:

Export Administrative Data

- **Admin** – Exports all administrative data to `admin.xml`, including data of the following types:
 - Group, GroupRegion, GroupRuleSet, GroupUser
 - Region
 - Role, Privileges, RolePrivilege, Permission
 - User, including AttributeUser, UserRole, UserSettings
 - UserPreference
- **Roles** – Exports all data that maps system permissions to roles to the file `roles.xml`. If you choose **Admin** as the export type, the same role data is exported with the other administrative data. The file has data of the following types:
 - Role
 - Privileges
 - RolePrivilege
 - Permission

Export security dictionary

You can export the security dictionary as a compressed XML or HTML file.

Importing and exporting either with Gosu or from the command prompt

You might want to import or export other types of data or use files in formats other than XML. For example, if you receive new information from an external system, you might want to import this new data into ContactManager in a single step. Gosu classes and command-prompt commands are your two alternatives to using the administrative user interface. Gosu classes enable you both to import and to export, but the command-prompt commands support only importing, not exporting.

Importing from the command prompt

There are command-prompt commands for importing, but not exporting, XML and CSV files containing any kind of data, not just the types of data supported by the **Administration** tab. See the *System Administration Guide*.

Importing and exporting with Gosu classes

You can import and export administrative data by using Gosu classes. See the *Integration Guide*.

Managing contact data

You can use Guidewire core application screens to manage contacts stored in ContactManager.

ClaimCenter

In the ClaimCenter **Address Book** tab, you can search for and view existing contacts. You can change ContactManager data in ClaimCenter by using screens for managing or adding claim contacts, such as the **New Claim** wizard or a claim's **Parties Involved** screen.

PolicyCenter

In the PolicyCenter **Contact** tab, you can create new contacts, search for existing contacts, select a recently viewed contact, and change contact information. You can also create an account for the contact. PolicyCenter additionally enables you to work with contacts in its **Account** and **Policy** screens, where you can add, remove, and update contacts in various roles.

BillingCenter

In the BillingCenter **Account** and **Policy** tabs, you can click **Contacts** to open the **Contacts** screen. On this screen you can add new contacts, search for existing contacts, select a recently viewed contact, and change contact

information. Additionally, you can search for contacts by clicking the **Search** tab and choosing **Contacts** from the drop-down list.

To work with ContactManager from a Guidewire core application, you must install both ContactManager and the Guidewire core application and integrate them.

You can also manage contacts stored in ContactManager by logging in to ContactManager as a user with the appropriate role. For example, you must log in to ContactManager to merge contacts or approve pending contact changes. Additionally, if you want to add contacts that are not on a claim, such as vendor contacts, you must do so in ContactManager.

You log in as a user with the Contact Manager role, which has permissions supporting viewing, searching for, merging, adding, editing, and deleting contacts and reviewing pending contacts. These permissions include:

- **View merge** permission, with code `abviewmerge`, which enables you to see **Merge Contacts** screens
- **View pending** permission, with code `abviewpending`, which enables you to see **Pending Contacts** screens

Other reasons to log in to ContactManager are to manage its users or to manage the server. For example, you can assign the User Admin role to a user. That user can then log in to ContactManager and manage its users.

See also

- “Installing ContactManager” on page 55
- “Integrating ContactManager with Guidewire core applications” on page 59.

Detecting and merging duplicate contacts

Because creating contacts can result in duplicate contact records, there are features in ContactManager to detect duplicate contacts and enable merging them. A user with appropriate privileges in ContactManager can run Duplicate Contacts Finder batch processing to detect duplicate contacts, and then merge each duplicate contact in the **Merge Contacts** screen.

IMPORTANT The first time you run Duplicate Contacts Finder batch processing, it can perform a large number of comparisons requiring considerable application resources. You can limit the number of contacts it processes by setting a processing time and date before which contacts are ignored. For more information, see “Configuring Duplicate Contacts Finder batch processing” on page 247. Additionally, Guidewire recommends that you set the work queue to run initially at a time when general access to your server is restricted. You might also need to allocate time for a ContactManager user to resolve what is likely to be a large number of duplicate contacts.

Configuring Duplicate Contacts Finder batch processing

Duplicate Contacts Finder batch processing compares a set of contacts that have recently changed against the rest of the contacts in the ContactManager database. *Recently* is defined by either of the following:

- Contact changes that occurred after the last time the batch process ran
- Contact changes that occurred after the date and time set in the configuration parameter `DuplicateContactsEarliestModificationDate`

For example:

1. Duplicate Contacts Finder batch processing ran today at 1:00 am. There were 100 contacts in the database at that time.
2. Duplicate Contacts Finder found 5 possible duplicate contacts.
3. The contact administrator checked today for duplicate contacts and merged 5 of them, leaving 95 contacts in the database.
4. After Duplicate Contacts Finder last ran at 1:00 am today, 10 new contacts were created.
5. Duplicate Contacts Finder runs at 1:00 am tomorrow. It compares each of those 10 new contacts against 104 contacts—the other 9 new contacts and the 95 contacts that were already in the database.

Duplicate Contacts Finder checks the value of `DuplicateContactsEarliestModificationDate`, which you can set in `config.xml`. Duplicate Contacts Finder ignores any contacts created or modified before the date and time in this

configuration parameter. You can use this setting to avoid processing every contact in the database the first time you run the batch process.

See also

“Set the DuplicateContactsEarliestModificationDate configuration parameter” on page 248

Set the DuplicateContactsEarliestModificationDate configuration parameter

Before you begin

Set this configuration parameter for Duplicate Contacts Finder. See “Configuring Duplicate Contacts Finder batch processing” on page 247.

About this task

For example, you imported a large number of contacts into the database on February 18, 2018, and you set DuplicateContactsEarliestModificationDate to 02/19/2018 12:00 AM. Duplicate Contacts Finder does not process any contacts from your February 18 import unless you modified them after 12:00 am on February 19, 2018. Duplicate Contacts Finder does process contacts modified after that date and time.

The date and time for this entry must have the following format:

```
mm/dd/yyyy hh:mm am
```

Procedure

1. Start Guidewire Studio™ for ContactManager.
At a command prompt, navigate to the ContactManager installation folder and enter the following command:

```
gwb studio
```

2. Navigate in the **Project** window to **configuration→config** and double-click config.xml to open this file in the editor.
3. Press Ctrl+F and enter DuplicateContactsEarliestModificationDate to find this parameter setting in the file.
4. Enter a date and time value.
DuplicateContactsEarliestModificationDate="02/19/2018 12:00 AM"
5. Save your changes.
6. If ContactManager is running, stop ContactManager and restart it to pick up the configuration change.
7. Run Duplicate Contacts Finder, or set it up to run automatically.

See also

- “Running the Duplicate Contacts Finder work queue manually” on page 248
- “Setting Duplicate Contacts Finder to run automatically” on page 249

Running the Duplicate Contacts Finder work queue manually

You can run Duplicate Contacts Finder batch processing manually in the **Batch Process Info** screen or from the command line. For example, you might run it to support testing of duplicate contacts.

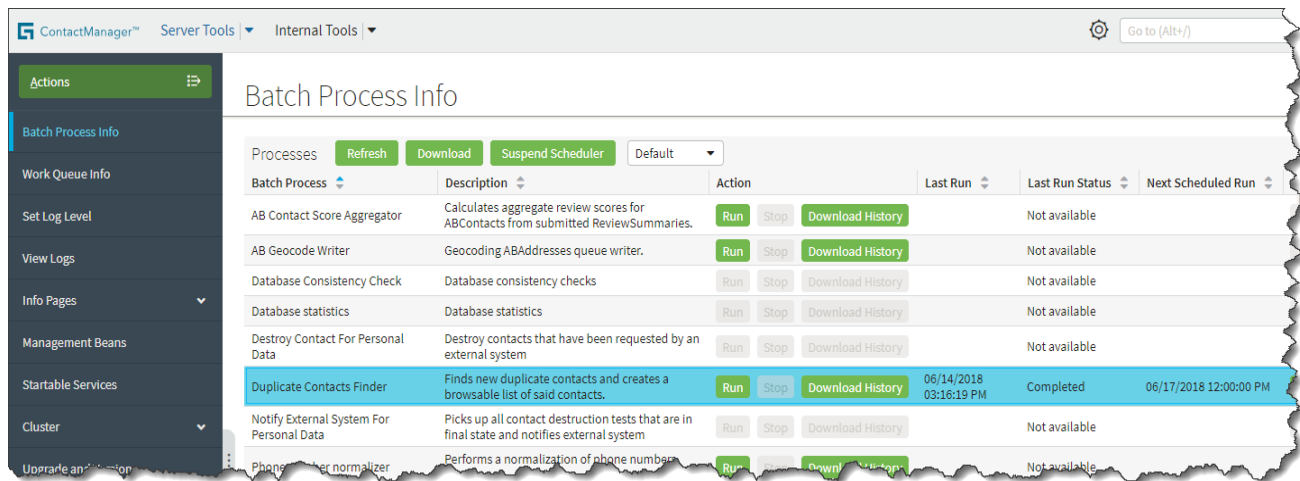
Run Duplicate Contacts Finder from the batch process info screen

About this task

You can run Duplicate Contacts Finder in the ContactManager user interface.

Procedure

1. Log in as a user with the Tools View role. This role has the permission `toolsBatchProcessview`, which enables you to work with work queues and batch processes. For example, log in with username `su` and password `gw`.
 2. Press `Alt+Shift+T` to open the screen showing the **Server Tools** tab and the **Internal Tools** tab.
 3. Click the **Server Tools** tab.
 4. From the Sidebar on the left under **Actions**, choose **Batch Process Info**.
 5. In the **Batch Process Info** screen, scroll down to **Duplicate Contacts Finder** and click the **Run** button in that row.
- When the process completes, the **Last Run Status** changes to **Completed** and the **Last Run** column updates to show the run date and time.



Run Duplicate Contacts Finder from the command line

About this task

You can run Duplicate Contacts Finder from the command line by using the `maintenance_tools` command.

Procedure

1. Open a command prompt and navigate to `ContactManager/admin/bin`.
2. Enter the following command:

```
maintenance_tools -startprocess duplicatecontacts
```

See also

- For information on the `maintenance_tools` command, see the *System Administration Guide*.

Setting Duplicate Contacts Finder to run automatically

In the base configuration, **Duplicate Contacts Finder** batch processing is set to run once a week on Sunday at 12:00 pm. You can:

- **Change this setting to run the work queue on a schedule** – See “Scheduling work queue writers” on page 250.
- **Set up the work queue itself** – See “Changing work queue settings” on page 250.
- **Limit the number of search results** – See “Changing match results and search scope settings” on page 251.
- **Set the scope of searches** – See “Changing match results and search scope settings” on page 251.

See also

- For information on administering batch processing and on work queues, see the *System Administration Guide*

Scheduling work queue writers

In Guidewire Studio™ for ContactManager, navigate in the **Project** window to **configuration→config→scheduler** and double-click `scheduler-config.xml` to open the file in the editor. In this XML file, you can change the settings for the `ProcessSchedule` named `DuplicateContacts`. The default entry sets the process to run at noon every Sunday.

```
<ProcessSchedule process="DuplicateContacts">
  <CronSchedule dayofmonth="?" dayofweek="SUN" hours="12"/>
</ProcessSchedule>
```

For example, the following entry would cause the process to run at midnight every night:

```
<ProcessSchedule process="DuplicateContacts">
  <CronSchedule hours="0"/>
</ProcessSchedule>
```

See also

- For complete information on the scheduling settings for work queue writers, see the *System Administration Guide*.

Changing work queue settings

In Guidewire Studio™ for ContactManager, navigate in the **Projects** window to **configuration→config→workqueue** and double-click `work-queue.xml` to open it in the editor. In this XML file, you can change the settings for the `workQueueClass` named `com.guidewire.ab.domain.contact.DuplicateContactsFinderWorkQueue`:

```
<work-queue
  progressinterval="600000"
  workQueueClass="com.guidewire.ab.domain.contact.DuplicateContactsFinderWorkQueue">
  <worker batchsize="100" instances="4"/>
</work-queue>
```

The settings are described in the comments at the beginning of the `work-queue.xml` file. The base configuration settings for `com.guidewire.ab.domain.contact.DuplicateContactsFinderWorkQueue` are:

minpollinterval="0"

The worker does not sleep after completing a work item.

maxpollinterval="60000"

The worker wakes up and polls for work every 60,000 milliseconds, or one minute.

progressinterval="600000"

The amount of time the system gives the worker to do the work before assuming that it ran into a problem and reassigning the work to another worker. By default this time is 600,000 milliseconds, or 10 minutes.

batchsize="100"

Each worker takes 100 new contact names at a time and compares each one to the contacts in the database.

instances="1"

The number of workers for the process.

See also

- For more information on work queues and work queue settings, see the *System Administration Guide*

Changing match results and search scope settings

You can set the maximum number of matches that can be found for a contact and how wide the search is. In Guidewire Studio™ for ContactManager, navigate in the **Project** window to **configuration**→**config** and open **config.xml**. Change the following parameters under the heading **DuplicateContactsFinderWorkQueue**:

- **MaxDuplicateContactsFinderWorkQueueResults** – By default, each **DuplicateContactsFinderWorkQueue** worker finds up to one thousand potential duplicates for a single contact. This parameter is intended to catch searches that have too loose a set of search fields and, therefore, match too many contacts in the database. If the worker finds more than the maximum allowed number of duplicates, it does not create the list of duplicates in its results list. Instead, it logs an exception that indicates the **LinkID** of the new contact. In addition, the new contact has an exception entry in the results list. The default setting is:

```
<param
  name="MaxDuplicateContactsFinderWorkQueueResults"
  value="1000"/>
```

- **DuplicateContactsWideSearch** – By default, this parameter is **true**, causing the process to perform a wide search using the match settings for **ABPerson**, **ABCompany**, and **ABPlace**. The default setting is:

```
<param
  name="DuplicateContactsWideSearch" value="true"/>
```

Note: This parameter applies only to duplicate finder batch processing. It does not set general searching behavior in the plugin **FindDuplicatesPlugin**.

When you specify wide search for duplicate finder processing, it can return matches for a larger number of subtypes than matching that uses **DEFAULT_MAP** (a setting of **false** for **DuplicateContactsWideSearch**). **WIDE_MAP** uses the contact's supertypes, **ABPerson**, **ABCompany**, and **ABPlace**. This type of search aids in finding a contact that was added erroneously to the database as more than one subtype.

In the base configuration, **DEFAULT_MAP** additionally returns results for **ABPersonVendor** and **ABCompanyVendor**, which are subtypes of **ABPerson** and **ABCompany**. For example, if the subtype of the contact being compared against the database is **ABDoctor**, a **DEFAULT_MAP** search returns only matches for the **ABPersonVendor** subtype. This subtype is the closest supertype specified in the map. The wide search in the base configuration returns matches for all **ABPerson** subtypes, not just **ABPersonVendor**.

The plugin **FindDuplicatesPlugin** defines the **ABContact** subtypes used in this search in the variable **WIDE_MAP**.

See also

- “**IFindDuplicatesPlugin** plugin interface” on page 289

Merge duplicate contacts

Before you begin

Merging duplicate contacts requires that **Duplicate Contacts Finder** batch processing has run. See:

- “**Configuring Duplicate Contacts Finder** batch processing” on page 247
- “**Running the Duplicate Contacts Finder** work queue manually” on page 248
- “**Setting Duplicate Contacts Finder** to run automatically” on page 249

After **Duplicate Contacts Finder** batch processing has run, you can see any potential duplicate contacts that this process found by clicking **Merge Contacts** on the **Contacts** tab.

Procedure

1. Log in as a user with the **Contact Manager** role.

For example, log in with username **su** and password **gw**.

The **Contact Manager** role includes the permission to view the merge screen, **abviewmerge**. To merge contacts, there are additional permissions to edit and delete contacts that are also part of the **Contact Manager** role, such as **abedit**, **abdelete**, and **anytagedit**.

2. Click the **Contacts** tab.
3. In the Sidebar, click **Merge Contacts**.
4. In the **Merge Contacts** screen, you see all the duplicate contacts detected by the last run of Duplicate Contacts Finder batch processing. The duplicates are grouped into pairs. You typically click **Review** to resolve one pair at a time, but you can mark multiple contact pairs in this screen and click **Ignore** for all of them.
 - You can select the check box next to one or more pairs that you determine are not duplicates and click **Ignore** to ignore all the checked pairs. Clicking **Ignore** saves both contacts in each pair and then removes each duplicate contact entry. These duplicate entries will not show up in future runs of the batch process unless a future edit makes them duplicates again.
 - You can search for specific duplicate contacts. If you do a search, you can use the **Match Type** list to filter the results by exact match, potential match, or all matches.
 - You might search for a specific contact's duplicates on this screen and then want to see all the duplicates again. To do so, click **Reset** and then click **Search**.
5. If you do not see any potential duplicate contacts listed, it is possible that Duplicate Contacts Finder has not run. It could also have run more than once since the last time you viewed the list.
 - If you clear the **Last Run Only** check box, you can see all duplicates that remain from any run of Duplicate Contacts Finder.
 - If that does not work, you need to run Duplicate Contacts Finder batch processing.
6. For any pair of contacts, if you click the **Review** button, you can compare the two contacts on the **Review Contacts for Merging** screen.

The **Review Contacts for Merging** screen has four actions you can perform on the two contacts you are comparing:

 - **Merge** – The contacts are duplicates. After you have completed all the comparisons on all tabs and the data is correct for both versions of the contact, click **Merge** to save the data in one contact. Clicking **Merge** removes the contact that is a duplicate and this duplicate contact entry. This duplicate entry will not show up in future runs of the batch process.
 - **Merge Then Edit** – The same as **Merge**, except that the merged contact opens in an editor after ContactManager completes the merge.
 - **Ignore** – The contacts are not duplicates. Clicking **Ignore** saves both contacts and removes this duplicate contact entry. This particular duplicate entry will not show up in future runs of the batch process unless a future edit makes them duplicates again.
 - **Cancel** – You do not want to decide if these two contacts are or are not duplicates. Clicking **Cancel** preserves this duplicate contact entry for this run of the batch process. However, the next run of the batch process will not pick up these duplicates. To see them in the **Merge Contacts** screens after any subsequent run of the batch process, you clear the **Last Run Only** check box, as described previously in Step 5.
7. On the **Review Contacts for Merging** screen, there are multiple tabs. Complete your work on all tabs before clicking **Merge** or **Merge Then Edit** at the top of the screen.
 - Initially you see the **Contact Detail** tab with four columns. The columns show the field names, the data for the contact to be kept, the data for the contact to be retired, and the data resulting from the merge.
 - The **Addresses** tab enables you to choose replacement addresses and to choose more than one address for the merged contact. You can also change the primary address and set the address type for each address. Check **Include** for any address that you want to keep that is not the primary address. If you do not include an address, you must indicate which address duplicates it in the address's **Duplicate Address** list. You can choose **None** from this list if the address is not duplicated by another address.
 - The **Documents** tab enables you to choose which documents to keep. Set **Include** for documents you want to be attached to the merged contact. Clear **Include** for documents you want to remove from the merged contact.
 - The **Related Contacts** tab enables you to choose which related contacts to keep.
 - The **EFT Information** tab enables you to choose which electronic funds transfer accounts to keep.
 - The **Vendor Data** tab is visible if one of the contacts has a vendor tag. This tab enables you to determine which tags, availability settings, and services to keep.
8. When the **Updated Contact** column has all the correct data, you can click **Merge** or **Merge Then Edit**.
 - **Merge** – Save the contact with the information you have chosen and exit the merge screens for this contact. ContactManager saves the **Kept** contact and retires the **Retired** contact. ContactManager then notifies all

integrated core applications that the contact has changed. On receiving this notification, the core applications can change references to the retired contact and make them references to the kept contact. See “Merging contacts and notifying core applications” on page 253.

- **Merge Then Edit** – Save the contact with the information you have chosen, and then open the **Kept** contact in the editor. ContactManager first saves the **Kept** contact and retires the **Retired** contact. When the editor opens, you see the newly merged data for the **Kept** contact in the editor. You can make further changes and save your changes.

ContactManager notifies all integrated core applications that the contact has changed. On receiving this notification, the core applications can change references to the retired contact and make them references to the kept contact. See “Merging contacts and notifying core applications” on page 253.

Merging contacts and notifying core applications

When you merge duplicate contacts, ContactManager keeps one contact and retires the other. As with any contact change, ContactManager then notifies the integrated core applications that a contact has been merged. ContactManager calls the core application implementation of the `ABClientAPI` interface method `mergeContacts` to notify the application of the kept contact and retired contact `AddressBookUID` values.

The application can then update the contact’s `AddressBookUID` with that of the kept contact. Each core application handles the change appropriately for the application. For example, the core application updates local copies of the retired contact to use the `AddressBookUID` of the kept contact. It then retrieves the data for the kept contact from ContactManager.

ContactManager provides two `ABContactAPI` web service methods, `getReplacementAddress` and `getReplacementContact`, that the core application can call to update the contact data by using the `AddressBookUID` of the kept contact.

In the base configuration, the core applications implement the `ABClientAPI` interface in the class `ContactAPI`. You can see how each application implements `mergeContacts` by opening the class in Guidewire Studio™ for that application. Each application uses a different path for the class:

BillingCenter

```
gw.webservice.bc.bc1000.contact.ContactAPI
```

ClaimCenter

```
gw.webservice.cc.cc1000.contact.ContactAPI
```

BillingCenter

```
gw.webservice.pc.pc1000.contact.ContactAPI
```

See also

- “Merge duplicate contacts” on page 251
- For a description of the method `ABClientAPI.mergeContacts`, see “`ABClientAPI` interface” on page 272.
- For descriptions of the methods `ABContactAPI.getReplacementAddress` and `ABContactAPI.getReplacementContact`, see “`ABContactAPI` methods” on page 264.

Review pending changes to contacts

About this task

Pending changes are saved in ContactManager when a ClaimCenter user who does not have `abcreate` or `abedit` permissions creates or edits a vendor contact. For a pending create or pending update to become a new contact or to update a contact, an authorized user must log in to ContactManager and approve it.

- For a pending create, ContactManager first creates the contact and then flags it as Pending Create. Approving the create simply removes the flag. Disapproving the create removes the pending contact.
- For a pending update, ContactManager stores the update information, but does not apply it to the contact until you approve it. If you disapprove the update, this information is deleted and is never applied to the contact.

To review pending changes to contacts

Procedure

1. Log in to ContactManager in a role that has permissions to create, update, view, delete, and search for contacts and to view pending contacts. For example, log in as a user with the Contact Manager role, such as the sample user `aaplegate/gw`.

For information on contact permissions and security, see “ContactManager contact security” on page 121.

2. Click the **Contacts** tab and then click **Pending Changes**.
3. On the **Pending Changes** screen, click either the **Updates** card to work with pending contact changes or the **Creates** card to work with new pending contacts.
4. Review each pending Update or Create request.
 - **Updates** – There is a list of contacts with information on the contact name, the ClaimCenter user who made the change, and the associated claim. When you select a contact, then, under **Entity or Property**, you can navigate to the fields that were changed. For each field, you can see the **Current Value** and the submitted pending change, the **New Value**.
 - **Creates** – There is a list of contacts with information on the contact name, the type of contact, the tags, the requesting user, and the claim number. When you select a contact, the contact details display below on the **Basics**, **Addresses**, and **Related Contacts** cards.
5. You can **Approve**, **Reject**, or **Approve Then Edit** the entire pending update or create for a contact. You can also click **Find Duplicates** for a pending create.

- **Reject** – If you reject the pending change or create, you see a screen asking for a reason and a text explanation. A reason is required. After selecting a reason and optionally entering an explanation, click **Reject Change** to complete the rejection.

ContactManager sends the rejection to ClaimCenter, which creates an activity for the user with the information you chose. ClaimCenter also changes the status message for the contact to indicate that the change did not go through.

- **Approve** – If you approve the change, the contact is updated or created, as needed.
- **Approve Then Edit** – If you want to make further changes to the contact data, click **Approve Then Edit**. The contact is updated or created, as needed, and then ContactManager opens the contact in an editor. At this point, the approved data is saved with the contact, and the contact is open in an editor so you can make further changes.

If the contact approval was for a pending update, ContactManager shows you both the old data and the new data in a worksheet below the **Edit** screen.

Note: If you cancel the edit or click **Return to Pending Updates**, you are effectively deciding not to edit the contact data. Canceling the edit is the same as simply approving the contact update. The old data is no longer available on the worksheet, but you can still access it. To see the old data, search for the contact and open its detail screen, and then click the **History** card to see the list of changes. For a particular change, click **Changes** to see the specifics.

- **Find Duplicates** – This button is available for a pending create only. If you want to check for duplicate contacts for the current contact that is pending creation, click **Find Duplicates**. If there are exact or potential matches, you see a list of the contacts in a worksheet. If one of the contacts is a duplicate of the pending create, you can click **Accept** for that contact.

ContactManager keeps the selected duplicate contact and retires the pending contact. ContactManager then sends the same message to ClaimCenter that it sends for any other duplicate contact merge. The message indicates the `AddressBookUID` of the retained contact and the `AddressBookUID` of the retired contact. ClaimCenter uses this information to update the link for its contact to the retained contact’s `AddressBookUID`.

Result

Note: If you do not check for duplicates and you approve creation of this contact, ContactManager does not automatically check for duplicate contacts for you. The new contact is created. If the new contact is a duplicate of an existing contact, you can make the correction in the **Merge Contacts** screens after Duplicate Contacts Finder batch processing runs. See “Merge duplicate contacts” on page 251.

Pending changes screen cache

The **Pending Changes** screen in ContactManager uses a cache for DiffDisplay objects to improve performance of this screen. These cached objects contain the display information for the changes being made as part of a pending update from an external system.

There are two configuration parameters for this cache in the `config.xml` configuration file, which you can edit in Guidewire Studio™ for ContactManager:

MaxDiffDisplaysInCache

Controls the maximum size of the cache. The default value is 60 entities.

DiffDisplaysCacheTimeoutInMinutes

Sets the amount of time since last access, in minutes, that a DiffDisplay object times out of the cache. The default value is 10 minutes.

The following rules remove entries in the cache:

- The rule Pending Contact Cache Update in the ABContactPreupdate rule set runs when an ABContact entity has changed. The rule removes entries in the cache for that entity.
- The rule Pending Contact Cache Update in the PendingContactChangePreupdate rule set runs when a pending change for an ABContact entity has been rejected. The rule removes entries in the cache for that entity.

See also

- “Review pending changes to contacts” on page 253
- “ABContact preupdate rule set” on page 213
- “PendingContactChangePreupdate rule set” on page 213

Working with work queues and batch processes

To get information about work queues and batch processes and start and stop them manually, use the **Batch Process Info** screen.

Start work queues and batch processes

Before you begin

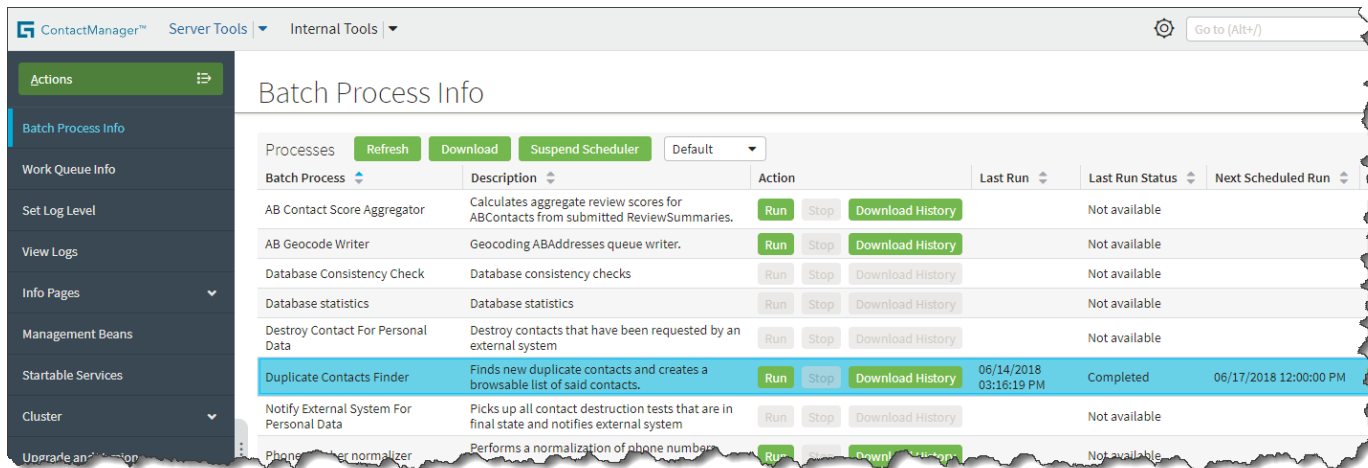
To start a work queue or batch process manually, you first log in to ContactManager as a user with an administrative role. The role must have the `toolsBatchProcessview` permission. See “Log in to ContactManager” on page 239.

Procedure

1. In ContactManager, press **Alt+Shift+T**.
2. Click the **Server Tools** tab.
3. In the Sidebar on the left, click **Batch Process Info**.

Batch Process Info screen

The following figure shows the **Batch Process Info** screen with **Duplicate Contacts Finder** selected:



The **Batch Process Info** screen enables you to run work queues and batch processes manually, without having to wait for the scheduled writers to run. You can start ContactManager processes and view information about them, including writers for work queues. This information includes the **Name**, **Description**, **Status**, **Last Run** time, **Last Run Status**, **Next Scheduled Run** time, and scheduling information.

To start a work queue or batch process manually, click **Run** in the **Action** column.

There are several batch processes you might find useful to run from this screen, especially during development.

Batch Process	Description
AB Contact Score Aggregator	Processes service provider review scores sent by ClaimCenter.
AB Geocode Writer	Geocodes addresses. For more information, see “Geocoding and proximity search for vendor contacts” on page 108.
Duplicate Contacts Finder	Compares new contacts to the existing contacts in the database to see if there are entries that might be duplicates of the contact. After Duplicate Contacts Finder runs, all new contacts that have been processed are marked to prevent them from being processed again. Possible duplicate contacts are saved to a list that a user with the appropriate permissions can process in the Merge Duplicates screen. For more information, see “Detecting and merging duplicate contacts” on page 247.

These batch processes are writers for work queues and have entries in the file `scheduler-config.xml` that you can uncomment and set to run at specific times.

See also

- “Start work queues and batch processes” on page 255
- “Scheduling work queue writers” on page 250
- “Schedule geocoding” on page 114.
- For information on administering batch processing, see the *System Administration Guide*

Improve query performance for large batch jobs

About this task

If you run a large batch job on more than ten percent of the addresses in the ContactManager database, query performance can be impacted. To improve query performance, after running the batch job you can run a series of database statistics statements from the command line.

Procedure

1. At a command prompt, navigate to `ContactManager/admin/bin`.
2. Run the following command:

```
maintenance_tools -password password -getdbstatisticsstatements | grep -i ab_abaddress > filename
```

In this command, *password* is your administration password and *filename* is the file where you want the output of the command to be saved. The `grep` command is available on UNIX or Linux systems. On Windows systems, you can run `grep` in a UNIX or Linux emulator like Cygwin.

3. Open the output file and run the database statistics statements in it.

See also

- For information on database statistics, see the *System Administration Guide*

ContactManager integration reference

Like the core Guidewire applications, ContactManager provides a set of web services, plugins, high level entities, mapping classes, and messaging events. ContactManager is intended to be integrated with the core applications, so it is important to understand the classes and files that enable ContactManager to communicate with the Guidewire core applications.

Overview of ContactManager integration

ContactManager, like the Guidewire core applications, is a complete application built on the Guidewire platform. ContactManager has its own versions of:

- Entities
- Rules
- Plugin interfaces
- Web service (WS-I SOAP) APIs published from the ContactManager application
- Messaging events
- Destination plugins
- Gosu API documentation (Gosudoc)

Integrating ContactManager with core applications from the InsuranceSuite—ClaimCenter, PolicyCenter, and BillingCenter—enables the core applications to use centralized contact management.

- In the ClaimCenter application, you can click the **Address Book** tab to search for and view contacts stored in ContactManager. You can add and edit contacts on a claim's **Contacts** screen and in the New Claim wizard.
- In PolicyCenter, you can click the **Contact** tab and create new contacts, search for existing contacts, select a recently viewed contact, and change contact information.
- In BillingCenter, you can click **Search**→**Contacts** to find contacts. You can also add, edit, and delete centrally managed contacts on **Contacts** screens of the **Account** and **Policy** tabs.

The Guidewire core applications send information to ContactManager across the network by using plugins to call ContactManager's web services. ContactManager itself has plugins that it uses to call the core applications' web services and to perform internal tasks.

Additionally, each core application implements a **ContactAPI** web service that ContactManager can use to send contact changes to the application.

To change or add properties to contact-related entities, you can extend the data model in both the Guidewire core applications and ContactManager. Alternatively, you can add tags for any contact, or you can specify services to use with vendor contacts. Whichever approach you take, you must ensure that information flows correctly between the applications.

The following areas of Guidewire core application functionality are not present in ContactManager:

- Notes
- Administrative groups
- Activities
- Assignment

There are no ContactManager entities, plugins, or APIs associated with these objects and features.

ContactManager entities

ContactManager has entities that are similar to entities in ClaimCenter, PolicyCenter, and BillingCenter. For example, the ClaimCenter, PolicyCenter, and BillingCenter Contact entity has a corresponding ContactManager ABContact entity.

ABContact is an important entity for ContactManager. It is the primary entity that ContactManager uses to manage contacts. The ABContact entity has the subtypes ABCompany, ABPerson, and ABPlace, and each of those three entities has specialized versions. For example, ABCompany has a vendor subtype called ABCompanyVendor.

This subtype hierarchy parallels the Contact subtype hierarchy in Guidewire core applications.

Note: ContactManager also has entities like User and Group to enable you to add employees who can access ContactManager. These entities are not involved in integrating with core applications.

Some important ContactManager entities are listed in the following table:

Entity or class	Description
ABContact	The ContactManager version of the Contact entity that stores name, phone number, address, and so on for the contact.
ABContactContact	The ContactManager version of the ContactContact entity that connects contacts that have a relationship.
ABContactTag	An entity connecting an ABContact entity to a tag, a typecode in the ContactTagType type-list. ABContact has an array of ABContactTag entities.
ABContactSpecialistService	An entity connecting an ABContact to a service. The ABContact entity has an array reference to this entity.
Address	An address associated with a contact.
Document	A document associated with a contact.
History	An entity that captures the history of actions performed on the contact, such as when it was created and what changes were made to the contact data. Additionally, this object can record which user and application performed the actions.
ReviewSummary	An entity that captures a summary of a review's information, passed from ClaimCenter.

See also

- “ABContact data model” on page 143
- “Contact data model” on page 145

ContactManager link IDs and comparison to other IDs

ContactManager entities that implement the ABLinkable delegate, such as ABContactTag, ABContactAddress, and ABContact and its subentities, have a LinkID property. This property uniquely identifies an ABLinkable entity instance for integration use, such as with Guidewire core applications. It is similar to the PublicID property in Guidewire core applications, which those applications can use as a primary key value for entities in external systems. However, because ContactManager is the system of record for contacts across the core applications, ContactManager must ensure that a contact or address be uniquely identifiable across all Guidewire applications. Therefore, unlike a PublicID, a LinkID cannot be changed.

If the core application does not specify an `External_UniqueID` when it calls `ContactManager` to create a new contact, `ContactManager` creates the `LinkID` for a new entity. If the core application does specify the unique ID when it sends a create request, `ContactManager` populates the `LinkID` with the `External_UniqueID` specified in the `XmlBackedInstance` data.

`ContactManager` passes the `LinkID` back to the core applications. A core application can use the return value either to identify the local version of the contact already created or to populate the equivalent `AddressBookUID` property.

Note: The unique ID passed to `ContactManager` for any `ABLinkable` entity in the `ABContact` graph might already exist on an entity of that type. If `ContactManager` detects that one of these entities, including retired entities, already use this ID, `ContactManager` throws a `DuplicateKeyException` for the call to `createContact`. In this case, the contact is not created in `ContactManager`. It is up to the calling application to recover from this state, either by providing a new unique ID or allowing `ContactManager` to create a unique ID.

The `AddressBookUID` property is the core application version of the `ContactManager` `LinkID` property. `AddressBookUID` properties and `LinkID` properties are mapped between core applications and `ContactManager` in the `ContactMapper` classes.

The following table compares the various types of IDs related to contacts:

Type of ID	Description
LinkID	The name for the internal ID that <code>ContactManager</code> uses for each <code>ABContact</code> entity and subentity
PublicID	The standard Guidewire public record ID that can be changed as needed.
AddressBookUID	In Guidewire core applications, this property of the <code>Contact</code> entity is the internal ID for a contact entity that is stored in <code>ContactManager</code> . If you are using a Guidewire core application and <code>ContactManager</code> together, an address book UID and a link ID have the same value. If you integrate with a different external contact management system, the address book UID has the same value as the internal ID of an object in that external application.

See also

- “`ContactMapper` class” on page 275
- For specifics on how `External_UniqueID` and `LinkID` are handled, see the `createContact` method in the table at “`ABContactAPI` methods” on page 264.
- For `ContactManager` examples of mapping between `AddressBookUID` and `LinkID`, see “Mapping fields of a `ContactManager` contact” on page 276.
- For examples of mapping between `AddressBookUID` and `LinkID`, see “Mapping fields of a core application contact” on page 279.
- For general information on `PublicID`, see the *Integration Guide*.

ContactManager web services

Web services provide a language-neutral, platform-neutral mechanism for invoking actions or requesting data from other applications across a network. Guidewire applications provide web services that are intended to be used both by other Guidewire applications and by external applications. You can also write your own web services.

See also

- For general, overview information on Guidewire web services, see the *Integration Guide*.

Web services provided by ContactManager

`ContactManager` provides the following web services. The first one, `ABContactAPI`, is the primary web service used by Guidewire core applications to communicate with `ContactManager`. The second one, `ABVendorEvaluationAPI`, supports vendor evaluations. The remaining web services are standard services available in all Guidewire applications.

For a complete list of base configuration web services, see the *Integration Guide*

Web Service	Description
ABContactAPI	The primary web service used by Guidewire core applications to search for, create, update, and delete contacts. This web service's WSDL is retrieved by every Guidewire core application to make it available to the application plugin that communicates with ContactManager. The class package is <code>gw.webservice.ab.ab1000.abcontactapi</code> . See "ABContactAPI web service" on page 264.
ABVendorEvaluationAPI	A web service that enables ClaimCenter to send and receive review summary information. The class package is <code>gw.webservice.ab.ab1000.abvendorevaluationapi</code> .
ImportToolsAPI	Imports administrative data from an XML file. Use this web service only with administrative database tables, such as entities of type User. The system does not perform complete data validation tests on any other type of imported data. For information on importing administrative data, see the <i>Integration Guide</i> .
LoginAPI	WS-I authentication happens with each API call. However, if you want to explicitly test specific authentication credentials in your web service client code, ContactManager publishes the built-in LoginAPI web service. Call this web service's login method, which takes a user name as a String and a password as a String. If authentication fails, the API throws an exception. You can also use LoginAPI to purposely leave a user session open for logging purposes. See the <i>Integration Guide</i> for more information on login authentication.
MaintenanceToolsAPI	Provides a set of tools that start and manage various background processes. The methods of this web service are available only when the server run level is maintenance or higher. See the <i>Integration Guide</i> for more information on the maintenance tools web service.
MessagingToolsAPI	Provides messaging methods for managing the messaging system remotely for recovery from message acknowledgment errors. The methods of this web service are available only when the server run level is multiuser. See the <i>Integration Guide</i> for more information on the messaging tools web service.
PersonalDataDestructionAPI	Enables an external application to request: <ul style="list-style-type: none"> • Destruction of a contact's data by AddressBookUID or by PublicID. • Destruction of a user by user name. See "Data destruction web service" on page 219.
ProfilerAPI	Sends information to the built-in system profiler. See the <i>Integration Guide</i> for more information on the profiling web service.
SystemToolsAPI	Provides a set of tools that are always available, even if the server is set to DBMaintenance run level. For servers in clusters, system tools API methods execute only on the server that receives the request. Some uses of this web service include: <ul style="list-style-type: none"> • Getting the version of the server, including application version and schema version • Checking the consistency of the underlying physical database • Getting and setting the run level See the <i>Integration Guide</i> for more information on the system tools web service.
TableImportAPI	Provides a table-based import interface for high volume data import. This web service is typically used for large-scale data conversions, particularly for migrating records from a legacy system into ContactManager prior to bringing ContactManager into production. See the <i>Integration Guide</i> for more information on the table import web service.
TypelistToolsAPI	Provides tools for getting the list of valid typecodes for a typelist and for mapping between ContactManager internal codes and codes in external systems. See the <i>Integration Guide</i> for more information on mapping typecodes and external system codes.
WorkflowAPI	Performs various actions on a workflow, such as suspending and resuming workflows and invoking workflow triggers. See the <i>Integration Guide</i> for more information on the workflow web service.

Web Service	Description
ZoneImportAPI	Imports geographic zone data from a comma separated value (CSV) file into a staging table, in preparation for loading zone data into the operational table. See the <i>Integration Guide</i> for more information on the zone data import web service.

Support classes for ContactManager web services

ContactManager provides a number of Gosu classes that are used by its web services. The following classes are the most common ones that you might edit to support your extensions to ContactManager, such as support for new search criteria. The class path for most of these classes is `gw.webservice.ab.ab1000.abcontactapi`.

Gosu Class	Description
ABContactAPIFindDuplicatesResult	A Gosu class used to define duplicate ABContact entities found by the ABContactAPI.findDuplicates method. You can edit this class to add your contact extensions so they can be returned by the findDuplicates method.
ABContactAPIPendingContactChange	An instance of this Gosu class is the second parameter to the method ABContactAPI.createContactPendingApproval. This class contains the metadata for the change that ContactManager will send back to the calling application if the change is rejected. The calling application can then notify the user of the rejection. See also “Review pending changes to contacts” on page 253.
ABContactAPIProximitySearchParameters	A Gosu class that defines the search parameters for a proximity search. You can modify this class. See also “Geocoding and proximity search for vendor contacts” on page 108.
ABContactAPISearchCriteria	A Gosu class that specifies the contact search criteria that the Guidewire core applications can use. The class package is <code>gw.webservice.ab.ab1000.abcontactapi</code> . This class is used by the web service ABContactAPI. You can edit this class to add new search criteria for contacts, as described in “Add search support in ContactManager for Guidewire core applications” on page 96.
ABContactAPISearchResult	A Gosu class that specifies the fields included in the contact search results that ContactManager sends back to the Guidewire core application. The class package is <code>gw.webservice.ab.ab1000.abcontactapi</code> . This class is used by the web service ABContactAPI. See “Add search support in ContactManager for Guidewire core applications” on page 96.
ABContactAPISpecialistService	A Gosu class that defines a SpecialistService entity to be returned by the method ABContactAPI.getSpecialistService. You can edit this class if you have made extensions to how Services work.
ABVendorEvaluationAPIReviewSummary	A Gosu class that enables ClaimCenter to update the review summary of a vendor evaluation with additional, new information. The class package is <code>gw.webservice.ab.ab1000.abvendorevaluationapi</code> . This class supports the web service ABVendorEvaluationAPI.

Gosu Class	Description
AddressInfo	A Gosu class that defines Address objects to be passed in ABContactAPI method calls. If you have extended the Address entity, you can edit this class to add your extensions, such as new address fields.

See also

- “ABContactAPI methods” on page 264

ABContactAPI web service

The ABContactAPI web service is the primary web service used by Guidewire core applications to search for, create, update, and delete contacts. This web service also supports specifying services in search criteria used in searches for contacts. This web service’s WSDL is retrieved by every Guidewire core application to make it available to the core application plugin that communicates with ContactManager.

- In ContactManager, ABContactAPI is in the class path `gw.webservice.ab.ab1000.abcontactapi`.
- In ClaimCenter, PolicyCenter, and BillingCenter, the WSDL for ABContactAPI is in the same web service collection for all three applications. Open Guidewire Studio™ and then open the **Project** window. Navigate to **configuration**→**gsrc** and then to `ws1/remote/gw/webservice/ab/ab1000.wsc`. Double-click `ab1000.wsc` to open the editor.

You can use this web service to load data from other data sources, to query contacts, to query services, or to update contacts. For example, you could write a command-prompt tool that enables an external system to add new contacts based on new business data from another part of the company.

ABContactAPI and typelists

The ABContactAPI web service exposes the typecodes in contact-related typelists as strings, which simplifies the code needed to access these typecodes from external or Guidewire core applications. For example, states are defined in the typelist `State`. The following annotation in ABContactAPI exposes this typelist as a string:

```
@WsiExposeEnumAsString(typekey.State)
```

Exposing this typelist as a string makes it possible for a Guidewire core application to do a simple assignment like the following in the ClaimCenter `ContactSearchMapper` class:

```
searchCriteriaInfo.ServiceState = searchCriteria.ServiceState.Code
```

ABContactAPI methods

The ContactManager ABContactAPI web service provides the following methods.

Method	Parameters	Description
createContact	abContactXML – Contact information in XmlBackedInstance format.	Creates a new contact and returns an AddressBookUIDContainer containing IDs for the Contact and child objects. Contact information is expected to be in XmlBackedInstance format. If the abContactXML parameter includes an External_UniqueID field and that field has a value, ContactManager uses this value to populate the LinkID field. In this manner, the calling application specifies the ultimate value of the LinkID.

Method	Parameters	Description
		<p>If the External_UniqueID field is missing or is null, ContactManager generates its own LinkID and passes it back to the calling application.</p> <p>ContactManager determines if the unique ID passed to ContactManager for any ABLinkable entity in the ABContact graph already exists on an entity of that type, including retired entities. If so, ContactManager throws a DuplicateKeyException for the call to createContact. In this case, the contact is not created in ContactManager. It is up to the calling application to recover from this state, either by providing a new unique ID or allowing ContactManager to create a unique ID.</p> <p>Calls ValidateABContactCreationPlugin to ensure that there is enough data to create the contact. If not, the method returns RequiredFieldException to the calling application.</p> <p>If there is enough data, the method creates a new ABContact of the subtype specified by abContactXML. The method then populates the new entity with data it retrieves by calling ContactIntegrationMapper.populateABContactFromXML.</p>
createContactPendingApproval	<p>abContactXML – Contact information in XmlBackedInstance format.</p> <p>updateContext – User, entity, and application information sent by core application. An instance of ABContactAPIPendingContactChange.</p>	<p>Creates a new contact of the type specified and sets its status to PENDING_APPROVAL. Returns an AddressBookUIDContainer containing IDs for the Contact and child objects and the update context and transaction ID.</p> <p>This method is called by the core application because the core application user creating the contact does not have permission to create a contact.</p> <p>Contact information is expected to be in XmlBackedInstance format.</p> <p>Calls ValidateABContactCreationPlugin to ensure that there is enough data to create the contact. If not, the method returns RequiredFieldException to the calling application.</p> <p>If there is enough data and no other exceptions are thrown, the method creates a new ABContact of the subtype specified by abContactXML. The method then populates the new entity with data it retrieves by calling ContactIntegrationMapper.populateABContactFromXML and sets its status to PENDING_APPROVAL.</p>
findDuplicates	<p>abContactXML – XmlBackedInstance that contains the contact data for which duplicates are being found.</p> <p>abContactAPISearchSpec – Specifies how the search is to be returned.</p>	<p>Finds contacts that match the specified contact.</p> <p>Returns an ABContactAPIFindDuplicatesResultContainer object containing summary information about each match.</p>

Method	Parameters	Description
<code>getReplacementAddress</code>	<code>addressLinkID</code> – linkID of an address that has been replaced because of a merge	Gets the address that has replaced the address passed in the parameter. An address can be replaced as a part of a merge. Returns the linkID of the address that replaces the address denoted by the argument.
<code>getReplacementContact</code>	<code>contactLinkID</code> – linkID of a contact that has been replaced because of a merge	Gets the contact that has replaced the contact passed in the parameter. A contact can be replaced as a result of a merge. Returns the linkID of the contact that replaces the contact denoted by the argument, or null if the contact has not been replaced by another contact.
<code>getSpecialistServices</code>	<code>contactLinkID</code> – linkID of the contact that has specialist services	Gets the specialist services associated with the contact passed in the parameter. Returns an array of <code>ABContactAPISpecialistService</code> objects, or null if the contact has no specialist services.
<code>removeContact</code>	<code>abContactXML</code> – <code>XmlBackedInstance</code> that contains the linkID of the contact to be removed and miscellaneous information.	Removes the contact with the matching linkID if the contact is not being used by any remote application. If a remote application is not running when this method runs, the contact is assumed to be in use and is not removed. Returns a <code>Boolean</code> indicating whether the contact was successfully removed.
<code>retrieveContact</code>	<code>linkID</code> – ID uniquely associated with this contact. In a core application, this value is the <code>AddressBookUID</code> .	Retrieves information about the contact uniquely specified by the linkID. Returns contact information in <code>XmlBackedInstance</code> format.
<code>retrieveDocumentsforContact</code>	<code>contactLinkID</code> – ID uniquely associated with this contact. In a core application, this value is the <code>AddressBookUID</code> . <code>abContactAPIDocumentSearchCriteria</code> – The criteria for the search. Required and cannot be null. <code>abContactAPIDocumentSearchSpec</code> – Specifications for how the results are to be returned.	Retrieves information about the contact's documents. Returns document information in <code>ABContactAPIDocumentSearchResultContainer</code> format.
<code>retrieveRelatedContacts</code>	<code>linkID</code> – ID uniquely associated with this contact	Retrieves information about the contact's related contacts. Returns a <code>RelatedContactInfoContainer</code> containing information about the contact's related contacts.

Method	Parameters	Description
	relationshipTypes – Array of ContactBidirectional relationship types of the related contacts to return information on. If null, then no restriction is enforced on the related contacts returned.	
searchContact	abContactAPISearchCriteria – Criteria for the search. This parameter must not be null. abContactAPISearchSpec – Specifies how the results are to be returned.	Searches for all contacts that match the given search criteria. Return an ABContactAPISearchResultContainer object containing the search results.
updateContact	abContactXML – Contact information in XmlBackedInstance format.	Updates an existing contact and returns an AddressBookUIDContainer object containing IDs for the Contact and child objects. Contact information is expected to be in XmlBackedInstance format. An existing ABContact is selected based on the abContactXML.LinkID. If none is found, the method throws BadIdentifierException. Then, origValue attributes for all fields and foreign keys being updated are checked against those same values in the selected ABContact. If discrepancies are found, the method throws EntityStateException. The purpose is to manage versioning of contact-related changes across multiple client applications. If no exceptions have yet been thrown, the method calls ContactIntegrationMapper.populateABContactFromXML to update the data for local entities from abContactXML.
updateContactPendingApproval	abContactXML – Contact information in XmlBackedInstance format. updateContext – User, entity, and application information sent by core application.	Submits for approval an update to an existing contact that is pending until approved. The core application calling this method has determined that the user updating the contact does not have permission to do so. Contact information is expected to be in XmlBackedInstance format. If no existing ABContact can be found based on the abContactXML.LinkID, the method throws BadIdentifierException. If an ABContact entity is found with this LinkID, this method creates a PendingUpdate entity for the contact.

Method	Parameters	Description
validateCreateContact	abContactXML – Contact information in XmlBackedInstance format.	Determines if the specified contact can be created. Calls <code>ValidateABContactCreationPlugin.validateCanCreate</code> to see if abContactXML has the minimum data required to create an ABContact entity. Returns an <code>ABContactAPIValidateCreateContactResult</code> object indicating whether validation passed and, if validation failed, an error message. For more information, see “ValidateABContactCreationPlugin plugin interface” on page 297.

Retrieving contacts and their relationships

If you want to integrate ContactManager with an application other than a Guidewire core application, you can write custom web services to do so. For example, to retrieve properties from contact records, you can write a custom web service (SOAP API) that extracts the subset of entity contact information that you want. Using an `AddressBookUID`, the API could extract properties from an `ABContact` record and return it from ContactManager to the SOAP client.

For typical cases, return only certain properties or subobjects, such as the properties needed to display or edit the record. Design your integration point accordingly to return only the necessary data, which reduces bandwidth and server resources.

In some cases, you might want to create new entities or classes to encapsulate your custom data.

If you are retrieving contact records, you can optionally retrieve related contacts, as you can do with ClaimCenter. Related contacts are referred to generally in ClaimCenter and ContactManager as *relationships*. For example, to retrieve a contact’s parent or guardian or employer contact information, specify one of these types of relationships. Finally, design web services for each integration point to return that information if it exists.

To use relationship retrieval, you must understand the difference between *source* and *target* relationships. These terms are ways of using the relationship identification codes for relationships such as *parent/guardian* and *employer*, but also specify the directionality of the relationship.

Relationship codes in Gosu are defined in the `ContactRel` typelist in Guidewire Studio™. For example, this Gosu enumeration includes relationship code `TC_employer` as a reference to the employer typecode in the typelist. Suppose you want the record for someone named John Smith. If you want to return John Smith’s employers, specify `TC_employer` as a *target relationship*. If you want to return John Smith’s employees, specify `TC_employer` as a *source relationship*.

A *target relationship* describes the relationship if the `ContactRel` typecode description fits it into the sentence with the following structure:

“I want to retrieve a contact and the _____ of the contact, if any is found.”

For example, for the employee relationship (`employer`), the sentence would be:

“I want to retrieve a contact and the **employer** of the contact, if any is found.”

If you want your request to specify the opposite of that relationship, such as employee instead of employer, specify the relationship as a source relationship.

While retrieving a contact record, you might want both directions of relationships. For example, you might want to simultaneously retrieve all employees of a company and the company’s primary contact. Use the `ContactBidiRel.EMPLOYER` source relationship and the `ContactBidiRel.PRIMARYCONTACT` target relationship.

IMPORTANT If extracting contact information by using custom SOAP APIs, write your API to return only the required properties and the required relationships. If you return an entire `ABContact` or unneeded related contacts, large amounts of data created by SOAP serialization can trigger server or client memory problems.

ABVendorEvaluationAPI web service

This web service provides methods used by ClaimCenter to communicate with ContactManager about service provider performance review information.

This web service provides the following methods:

Method	Parameters	Description
<code>addNewReviewSummary</code>	<code>reviewInfo</code> – The new review summary, an <code>ABVendorEvaluationAPIReviewSummary</code> object. The review summary sent to <code>ContactManager</code> must not have a <code>LinkID</code> set.	Adds a new review summary based on the <code>ABVendorEvaluationAPIReviewSummary</code> object passed in. Returns the created review summary, an <code>ABVendorEvaluationAPIReviewSummary</code> object, complete with <code>LinkID</code> .
<code>deleteReviewSummary</code>	<code>linkID</code> – The <code>LinkID</code> of the review summary as a <code>String</code> .	Deletes the review summary that has the passed-in <code>LinkID</code> . Returns a <code>Boolean</code> indicating if the review summary was successfully deleted.
<code>updateReviewScoresForContact</code>	<code>linkID</code> – The <code>LinkID</code> of the associated vendor contact, as a <code>String</code> .	Updates the scores for all the reviews for the contact whose <code>LinkID</code> is passed in. Returns an <code>int</code> indicating the current score.

ContactManager messaging events

`ContactManager` sends messages to integrated Guidewire core applications after changes to certain entities, such as adding or deleting a contact or changing data for a contact. These changes trigger events, which trigger code that sends messages. For example, a user of a core application changes the address of a contact stored in `ContactManager`. `ContactManager` receives that change and updates the contact, triggering an event, `ABContactChanged`. This event triggers event message rules that cause `ContactManager` to send the contact change to the integrated core applications.

You can see the messaging events that are generated for each entity in the *Data Dictionary*. To build the data dictionary for your system, open a command prompt and navigate to the `ContactManager` installation folder, and then enter the command `gwb genDataDictionary`. The command builds the security and data dictionaries. It saves the *Data Dictionary* in `ContactManager/build/dictionary/data/index.html`.

ContactManager messaging events by entity

The following table lists the messaging events for each `ContactManager` entity for which `ContactManager` sends messages.

Entity	Events	Description
ABContact	ABContactAdded ABContactChanged ABContactRemoved	Contact entities exist only as subtypes of <code>ABContact</code> , such as <code>ABPerson</code> . Those subtypes generate standard A/C/R events for root entity <code>ABContact</code> .
	ABContactPendingChangeRejected	An <code>ABContact</code> pending change request has been rejected.
	ABContactResync	An <code>ABContact</code> has been resynchronized. Re-send all related messages to external systems as appropriate.
		<p>See also</p> <ul style="list-style-type: none"> For information on message ordering and multi-threaded sending, see the <i>Integration Guide</i>.
ABAdjudicator	ABContactAdded ABContactChanged ABContactRemoved	Standard A/C/R events for root entity <code>ABContact</code> .

Entity	Events	Description
	ABContactPendingChangeRejected ABContactResync	
ABAttorney	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABAUTORepairShop	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABAUTOtowingAgcy	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABCompany	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABCompanyVendor	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABDoctor	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABLawFirm	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABLegalVenue	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABMedicalCareOrg	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABPerson	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.

Entity	Events	Description
ABPersonVendor	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABPlace	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABPolicyCompany	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABPolicyPerson	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.
ABUserContact	ABContactAdded ABContactChanged ABContactRemoved ABContactPendingChangeRejected ABContactResync	Standard A/C/R events for root entity ABContact.

See also

- For information on ContactManager event messaging rules, see “EventMessage rule set category” on page 209.

ABContact message safe ordering

ContactManager supports safe ordering of Message entity instances associated with ABContact entity instances. Messages associated with an entity instance are sent ordered by ABContact instance for each messaging destination. Safe ordering allows only one message per contact-destination pair to be *in flight*—sent but not acknowledged at any given time. Messages of this type are referred to as *safe-ordered messages*.

If an event generates two safe-ordered messages related to one entity instance, ContactManager sends the first message immediately but does not send the second message. After ContactManager receives an acknowledgment from the destination about the first message, it sends the second message.

Because ContactManager manages events for many entity instances, a Guidewire core application can have many messages in flight at a time, even multiple messages to one destination. However, there can be only one message in flight for each contact-destination pair.

See also

ABContact message resynchronization

ContactManager supports resynchronizing a contact—the ABContact entity—with an external system. If an ABContact entity resynchronizes, a destination could listen for the ABContactResync event. If that event triggers, the Event Fired rules that process it could re-send important messages to external systems for this entity to synchronize with the external system.

ABClientAPI interface

ContactManager requires that each Guidewire core application implement the ABClientAPI interface and expose it as a web service.

- ClaimCenter implements this interface in `gw.webservice.cc.cc1000.contact.ContactAPI`.
- PolicyCenter implements this interface in `gw.webservice.pc.pc1000.contact.ContactAPI`.
- BillingCenter implements this interface in `gw.webservice.bc.bc1000.contact.ContactAPI`.

ContactManager then calls the web service methods in each application when it needs to broadcast changes in contacts to the applications. Each Guidewire core application can determine what to do in response to the call from ContactManager.

The current version of this ContactManager interface is in the package `gw.webservice.contactapi.ab1000.ABClientAPI`.

See also

- For information on core application implementations of ContactAPI, see the contact web service topics in the *Integration Guide*.

Method signatures of the ABClientAPI interface

The ABClientAPI interface provides the following method signatures.

Method	Parameters	Description
<code>isContactDeletable</code>	<code>addressBookUID</code> – The address book unique ID of the contact. For ContactManager, this is the <code>linkID</code> .	Return true either if the contact associated with the <code>AddressBookUID</code> can be deleted or if no contact is associated with <code>AddressBookUID</code> . Return false if the contact cannot be deleted.
<code>mergeContacts</code>	<code>keptContactABUID</code> – The addressBookUID of the contact to be kept. <code>deletedContactABUID</code> – The addressBookUID of the contact to be deleted.	Merge two contacts that were merged in ContactManager. ContactManager does not send contact data with this method call. It is up to the Guidewire core application to retrieve the data for the kept contact.
<code>pendingCreateApproved</code>	<code>context</code> – An <code>ABClientAPIPendingChangeContext</code> object providing information on the user requesting this change.	Notifies the client system that a pending contact creation it submitted has been approved by ContactManager. The client application can use the information in the context parameter to inform the user who submitted the request that the contact was created. Additionally, the core application can update the sync status of the contact and post an appropriate message.
<code>pendingUpdateApproved</code>	<code>context</code> – An <code>ABClientAPIPendingChangeContext</code> object providing information on the user requesting this change.	Notifies the client system that a pending update it submitted has been approved by ContactManager. The client application can use the information in the context parameter to inform the user who submitted the change that the change was approved. Additionally, the core application can update the sync status of the contact and post an appropriate message.

Method	Parameters	Description
<code>pendingCreateRejected</code>	<code>context</code> – An <code>ABClientAPIPendingChangeContext</code> object providing information on the user requesting this change.	Notifies the client system that a pending contact creation it submitted has been rejected by <code>ContactManager</code> . The client application can use the information in the context parameter to inform the user who submitted the contact to be created that the creation was rejected. Additionally, the core application can update the sync status of the contact and post an appropriate message.
<code>pendingUpdateRejected</code>	<code>context</code> – An <code>ABClientAPIPendingChangeContext</code> object providing information on the user requesting this change.	Notifies the client system that a pending update it submitted has been rejected by <code>ContactManager</code> . The client application can use the information in the context parameter to inform the user who submitted the change that the change was rejected. Additionally, the core application can update the sync status of the contact and post an appropriate message.
<code>removeContact</code>	<code>addressBookUID</code> – Unique ID of a contact that has been removed in <code>ContactManager</code> .	The contact with this <code>AddressBookUID</code> has been removed in <code>ContactManager</code> . This call does not require that the contact be removed by the Guidewire core application. For example, if the contact is in use by an account, <code>PolicyCenter</code> does not retire it.
<code>updateContact</code>	<code>contactXML</code> – <code>XmlBackedInstance</code> that contains the <code>addressBookUID</code> of the contact and the updates to be made.	Data for this contact has changed in <code>ContactManager</code> . Update the contact by using the data in <code>contactXML</code> .

Deleting contacts in ClaimCenter

`ClaimCenter`, unlike the other core applications, can have multiple local instances of any contact, an instance for each claim. Therefore, `ClaimCenter` might have a lot of work to do in deleting a contact. `ClaimCenter` has to check every local instance of the contact. For each local instance, `ClaimCenter` must ensure that there are no entities that reference the instance that would prevent it from being deleted. For example, if there are any claims that have a contact as an involved party, the contact cannot be deleted.

See also

- “Synchronizing `ClaimCenter` and `ContactManager` contacts” on page 203
- For information on `Contact` web service APIs, see the *Integration Guide*.

ClaimCenter ContactAPI web service methods for removing contacts

The `ClaimCenter` implementation of `ABClientAPI` is the web service `gw.webservice.cc.cc1000.contact.ContactAPI`. This web service provides implementations of all the methods, two of which are used by `ContactManager` in requesting that `ClaimCenter` delete a contact:

`isContactDeletable(addressBookUID : String)`

If there are fewer than ten local instances of the contact, this method finds all the contacts linked to the `addressBookUID`. It then calls the `ContactRetireHelper.retireContact(Contact)` method on each one and returns `true` if they can all be successfully retired.

If any contact cannot be successfully retired, the method returns `false` to indicate that the contact cannot be deleted. If there are more than ten contacts, the method returns `false` and then generates work items to check all the local contacts as part of the work queue. The work queue calls `ContactRetireHelper.retireContact` to retire each contact instance.

removeContact(addressBookUID : String)

Calls `ContactRetireHelper.retireContact` for each ClaimCenter contact found with the `addressBookUID`.

ClaimCenter classes for removing contacts

ClaimCenter provides two classes you can use in configuring how ClaimCenter handles any data model extensions you have added that might affect the removal, or *retiring*, of contacts.:

IRetireContactPlugin

This plugin interface has a method to return a set of safe properties. These properties represent foreign key links to a contact that can be retired without any further checking when a contact is retired. You can add to the list any extension foreign keys that you deem safe. You can use the implementation of this interface, `gw.plugin.contact.RetireContactPlugin`, to add properties to the safe list.

ContactRetireBean

This interface can be implemented by entities that have an unsafe foreign key link to a contact. The implementation can determine if an entity instance will prevent a contact instance from being retired. If not, the contact can be retired, and then the implementation can also determine if any other contacts or entities can be retired along with the contact. See the class `gw.api.contact.ContactContactRetireBeanImpl` for an example of checking to see if a `ContactContact`, and the associated `Contact`, can be retired.

Implementation details for retiring contacts in ClaimCenter

ClaimCenter provides the `gw.api.contact.ContactRetireHelper` class to determine if a contact can be retired. This class provides two methods used in retiring contacts:

public static boolean retireContact(Contact contact)

Attempts to retire the passed in `Contact`, returning true if it was successful in doing so. This method is called by the `ContactRetire` work queue.

public static boolean computeCanRetireContact(Contact contact, ContactRetireContext retireContext)

Available to be used when implementing the `ContactRetireBean` interface to see if a contact can be retired. You might call this method if, while using `ContactRetireBean.computeCanRetireBeanForContactProperty`, you encounter another contact and need to retire it as well.

The `retireContact` method splits the foreign key references for the `Contact` into two components, those properties deemed safe from the `IRetireContactPlugin` implementation and the other properties. The safe properties for the contact do not block the retirement of the contact. Additionally, if there is a bean connected to the contact through this property, it is retired along with the contact.

In the base implementation, the safe references are:

Entity	Array field
ContactAddress	Contact
ContactCategoryScore	Contact
ContactTag	Contact
EFTData	Contact
OfficialID	Contact
Review	Contact

The other properties by default will block the retirement of the contact unless they implement the `ContactRetireBean` interface. If an entity implements that interface, the `computeCanRetireBeanForContactProperty` method of the interface can be called to see if the entity can be retired.

In the base configuration, the only entity requiring this interface to be implemented is `ContactContact`. This entity has the dual role of being an array on `Contact` and also having a foreign key to a contact. `ContactContact` represents a join table to two contacts, so the other contact is a foreign key reference. The class that implements the `ContactRetireBean` interface is `gw.api.contact.ContactContactRetireBeanImpl`.

ContactMapper class

Guidewire core applications send contact information to `ContactManager` by using an XML SOAP object. They use the `ContactMapper` class both to generate the SOAP object and to interpret it when they receive an XML SOAP object from `ContactManager`.

`ContactManager` and the Guidewire core applications each have their own version of the `ContactMapper` class. This class enables the applications to convert `Contact` entities and subentities to XML and send them to `ContactManager`. The class also enables Guidewire core applications to receive XML representations of the entities from `ContactManager` and convert them to `Contact` entities.

`ContactManager` also uses its version of this class to generate XML to send to the applications and to interpret the XML it receives from the applications. The class file in each core application and in `ContactManager` explicitly specifies every field of the entities to send or receive and how to handle them in that application.

In the base configuration of each Guidewire core application, the `ContactMapper` class is configured to handle all the `Contact` entities and subentities. The class handles the entities and fields of those entities that are to be sent to and received from `ContactManager`.

In the base configuration, there are differences in entity names, typecodes, and contact entity field names between the core applications and `ContactManager`. Each Guidewire core application has its own domain namespace, but the `ContactManager` web services require the `ContactManager` domain namespace. The core applications use `ContactManager` web services to communicate with `ContactManager`. Therefore, when there are differences in names of fields, data types, and typecodes between the core application and `ContactManager`, each core application is responsible for doing the name mapping.

- To map differing entity field names, a core application uses a method on the `ContactMapper` class.
- To map names of data types and typecodes, a core application uses the `ContactMapper` class in conjunction with a mapping class that specifies these name mappings. For example, a `Contact` in the core application maps to an `ABContact` in `ContactManager`.

The `ContactManager` `ContactMapper` class handles only `ABContact` entities and subentities. Because `ContactManager` must be able to communicate with all the Guidewire core applications, its `ContactMapper` class must contain all fields of each type of contact that each core application needs. In a Guidewire core application, you can map just the contact fields that you want to send to `ContactManager` and receive from it.

The `ContactMapper` class creates a SOAP object called `XMLBackedInstance` to pass the data between a Guidewire core application and `ContactManager`. This object contains the fields and data of contacts and their related arrays and foreign keys that are defined in the `ContactMapper` class. `XMLBackedInstance` has a representation both as an object and as an XML string.

If you are working only with the entities supplied by Guidewire in the base configuration, you do not need to make any changes to the `ContactMapper` class.

You can extend your Guidewire core application's `Contact` data model. You make the extension in `ContactManager` as well, and then map the entities to each other in `ContactMapper` in both the Guidewire core application and in `ContactManager`. Additionally, you use the application's mapping class to map differing entity names and any different typecodes to and from `ContactManager`.

See also

- “Core application mapping” on page 151.
- For information on the contact data model in the base configuration, see “Overview of contact entities” on page 143.
- “Extending the contact data model” on page 143.

ContactManager ContactMapper class

In ContactManager, you use the class `gw.contactmapper.ab1000.ContactMapper` to map ABContact entities and subentities. The class maps them to an XML object to send to Guidewire core applications and maps them from the XML objects received from Guidewire core applications. The class in the base ContactManager configuration works only with ABContact entities and subentities. ContactManager leaves it to the applications to map ABContact entities and subentities to and from Contact entities and subentities.

You can access this class in Guidewire Studio™ for ContactManager in the **Project** window. Navigate to **configuration**→**gsrc** and then to `gw.contactmapper.ab1000.ContactMapper`.

The following methods map the fields, foreign keys, and arrays of the ContactManager ABContact entities and subentities:

fieldMapping

Maps fields of an entity

fkMapping

Maps foreign keys of an entity

arrayMapping

Maps array references of an entity

Mapping fields of a ContactManager contact

The method `fieldMapping(Entity#Field)` by default maps ABContact entity and subentity fields in both directions, both to and from a Guidewire core application.

Note: Parameters to methods in ContactMapper use Gosu *feature literals* syntax to statically refer to an entity's fields. See the *Gosu Reference Guide*.

The mapping directions are:

To a Guidewire core application

From a ContactManager entity to an XML object to be sent to a Guidewire core application

From a Guidewire core application

From an XML object sent by a core application to a ContactManager entity

You also use this method to map fields of an entity that the contact references with either an array reference or a foreign key. To map a foreign key or array reference itself, you use different methods.

Each complete method call must be followed by a comma unless it is the last method call in the block.

Specifying a Single Direction

You specify a single direction by using `.withMappingDirection`, as follows:

.withMappingDirection(TO_XML)

Maps the field from ContactManager to the core application.

For example, use this method call to map a LinkID field to a core application:

```
fieldMapping(ABContact#LinkID)
    .withMappingDirection(TO_XML),
```

.withMappingDirection(TO_BEAN)

Maps the field from the core application to ContactManager.

Mapping externally specified unique IDs of a ContactManager contact

ContactManager supports creation of a contact with an external unique ID specified by the core application.

- If an external unique ID is specified in the `XmlBackedInstance` data sent in the `createContact` method call, ContactManager populates the `LinkID` of the new contact with that value. The field that ContactManager checks for in the `createContact` call is `External_UniqueID`.
- If the `createContact` method call does not specify an external unique ID, ContactManager generates its own unique ID and stores that value in the contact's `LinkID`.

For example, the following lines of mapping code from various parts of the `construct` method in the ContactManager class `ContactMapper` support this mechanism:

```
fieldMapping(ABContact#External_UniqueID),
fieldMapping(ABContactAddress#External_UniqueID),
fieldMapping(Address#External_UniqueID),
fieldMapping(ABContactTag#External_UniqueID),
fieldMapping(EFTData#External_UniqueID),
fieldMapping(ABContactCategoryScore#External_UniqueID),
```

See also

- “PolicyCenter mapping of externally specified unique IDs” on page 281
- “Creating and linking a contact” on page 195
- “ContactManager link IDs and comparison to other IDs” on page 260
- “ABContactAPI methods” on page 264

Mapping foreign keys of a ContactManager contact

The method `fkMapping(Entity#ForeignKey)` maps foreign keys for `ABContact` entities, subentities, and join tables. You can use the following additional qualifying methods as well:

withMappingDirection(TO_XML)

Specifies that the foreign key is mapped to the core application.

withMappingDirection(TO_BEAN)

Specifies that the foreign key is mapped from the core application.

Use the `fieldMapping` method to map all the fields of the entity to which a foreign key refers.

For example, the foreign key on `ABContactAddress` that points to an `Address` object is defined as follows:

```
fkMapping(ABContactAddress#Address),
```

Additionally, the fields for the `Address` object must also be defined by using `fieldMapping` method calls. The code for the `Address` entity referenced by the foreign key is:

```
fieldMapping(Address#LinkID)
    .withMappingDirection(TO_XML),
fieldMapping(Address#External_PublicID),
fieldMapping(Address#AddressLine1),
fieldMapping(Address#AddressLine1Kanji),
fieldMapping(Address#AddressLine2),
fieldMapping(Address#AddressLine2Kanji),
fieldMapping(Address#AddressLine3),
fieldMapping(Address#AddressType),
fieldMapping(Address#City),
fieldMapping(Address#CityKanji),
fieldMapping(Address#Country),
fieldMapping(Address#County),
fieldMapping(Address#Description),
fieldMapping(Address#GeocodeStatus),
fieldMapping(Address#PostalCode),
fieldMapping(Address#State),
fieldMapping(Address#ValidUntil),
```

```
fieldMapping(Address#CEDEX),
fieldMapping(Address#CEDEXBureau),
```

Mapping array references of a ContactManager contact

The method `arrayMapping(Entity#ArrayKey)` maps array references for `ABContact` entities and subentities. You can use the following additional qualifying methods as well:

`withMappingDirection(TO_XML)`

Specifies that the array reference is mapped to the core application.

`withMappingDirection(TO_BEAN)`

Specifies that the array reference is mapped from the core application.

Use the `fieldMapping` method to map all the fields of the entity to which an array reference refers.

For example, the array extension example “Extending contacts with an array” on page 171 adds a new entity named `ContactServiceState`. This entity represents a state in which a vendor contact operates. Each vendor can operate in more than one state, so the example adds an array reference to `ABContact` named `ContactServiceArea` that references an array of `ContactServiceState` entities.

Note: The example also creates a `ClaimCenter ContactServiceState` entity and a `ContactServiceArea` array reference on the `ClaimCenter Contact` entity.

To transfer this data into and out of `ContactManager`, you define the array reference and map the fields of the entity to which it refers in `ContactMapper`, as follows:

```
arrayMapping(ABContact#ContactServiceArea),
fieldMapping(ContactServiceState#LinkID)
    .withMappingDirection(TO_XML),
fieldMapping(ContactServiceState#External_PublicID),
fieldMapping(ContactServiceState#External_UniqueID),
fieldMapping(ContactServiceState#ServiceState)
```

Special handling in ContactManager for addresses

Addresses sent from core applications require special handling in `ContactManager` to handle scenarios like swapping a primary address for a secondary address. Turning address data into XML does not require any special handling, so addresses being sent to core applications use the normal `ContactMapper` code. For these reasons, the following foreign key and array references are defined as one-way, from `ContactManager` to the core application:

```
fkMapping(ABContact#PrimaryAddress)
    .withMappingDirection(TO_XML),
arrayMapping(ABContact#ContactAddresses)
    .withMappingDirection(TO_XML),
```

When a change to a primary or secondary address for a contact comes in from a core application, `ContactManager` calls `ABUpdateBeanPopulator.populateAddresses` instead of the regular mapping code.

While the address foreign key and array reference require special handling, the fields of an address do not. You can add or delete address fields as needed by using the `fieldMapping` method.

Core application ContactMapper class

In the Guidewire core applications, you use the class `gw.contactmapper.ab1000.ContactMapper` to map entities between the core application and `ContactManager`. The class maps entities to an XML object to send to `ContactManager` and maps entities from the XML objects received from `ContactManager`.

You can access this class in Guidewire Studio from the **Project** window. Navigate to **configuration**→**gsrsrc** and then to `gw.contactmapper.ab1000.ContactMapper`.

Note: ContactManager also has a ContactMapper class. If you want ContactManager to handle changes you make in the ClaimCenter class, you must update the ContactManager class as well. For example, you might add a new field to a Contact subtype and want ContactManager to use that new field with the ABContact subtype.

Each core application uses ContactMapper to map differing field names used by the core application and ContactManager. To map differing contact entity names and typecodes used by ClaimCenter and ContactManager, each core application uses a separate XXNameMapper Gosu class. You can access this class for each core application in Guidewire Studio from the **Project** window. Navigate to **configuration→gsrc** and then to `gw.contactmapper.ab1000`. Double-click the following class for your application to open it in an editor:

- ClaimCenter – CCNameMapper
- PolicyCenter – PCNameMapper
- BillingCenter – BCNameMapper

There is no equivalent ContactManager class for mapping differing entity and typecode names. All the mapping of differing names is done on the core application side.

See also

- “ContactManager ContactMapper class” on page 276.
- “ContactMapper class” on page 275.
- For an example that shows how to map entity names from ClaimCenter to ContactManager, see “Map the new subtype names in ClaimCenter” on page 160.

Mapping fields of a core application contact

The method `fieldMapping(Entity#Field)` by default maps Contact entity and subentity fields in both directions:

To ContactManager

From a core application entity to an XML object to be sent to ContactManager

From ContactManager

From an XML object sent by ContactManager to a core application

You also use this method to map fields of an entity that the contact references with either an array reference or a foreign key. To map a foreign key or array reference itself, you use different methods, as described later.

Each method call with qualifying methods, if any, is an entry in the set of return values. If you add a method to the set, add a comma after it if your method is not the last method in the set.

Specifying a single mapping direction for a field

You specify a single direction for the field mapping by using the qualifying method `withMappingDirection`, as follows:

`.withMappingDirection(TO_XML)`

Maps the field from the core application to ContactManager.

For example, use the following method call to map a `PublicID` field to the ContactManager `External_PublicID` field. This mapping is one-way because the core application sets and maintains this value.

```
fieldMapping(Contact#PublicID)
    .withMappingDirection(TO_XML)
    .withABName(MappingConstants.EXTERNAL_PUBLIC_ID),
```

`.withMappingDirection(TO_BEAN)`

Maps the field from ContactManager to the core application.

Mapping fields with differing names

If a field for a contact has a different name in the core application from the name in ContactManager, you map it by using the `withABName` method. For example, the `AddressBookUID` property on a ClaimCenter Contact entity is

called `LinkID` on an `ABContact` in `ContactManager`. The following code maps these different field names for a `Contact` entity:

```
fieldMapping(Contact#AddressBookUID)
    .withABName(MappingConstants.LINK_ID),
```

Mapping foreign keys of a core application contact

The method `fkMapping(Entity#ForeignKey)` maps foreign keys for `Contact` entities, subentities, and join tables. The method by default operates in both directions. You can use the following additional qualifying methods as well:

`withMappingDirection(TO_XML)`

Specifies that the foreign key is mapped to `ContactManager`.

`withMappingDirection(TO_BEAN)`

Specifies that the foreign key is mapped from `ContactManager`.

`withABName(name:String)`

Specifies a different name for the foreign key on the entity in `ContactManager`

Use the `fieldMapping` method to map all the fields of the entity to which a foreign key refers.

For example, the foreign key on `ContactAddress` that points to an `Address` object is defined as follows:

```
fkMapping(ContactAddress#Address),
```

Additionally, the fields for the `Address` object must all be defined by using `fieldMapping` method calls. The code for the `Address` entity referenced by the foreign key is:

```
fieldMapping(Address#AddressBookUID)
    .withABName(MappingConstants.LINK_ID),
fieldMapping(Address#PublicID)
    .withMappingDirection(TO_XML)
    .withABName(MappingConstants.EXTERNAL_PUBLIC_ID),
fieldMapping(Address#AddressLine1),
fieldMapping(Address#AddressLine2),
fieldMapping(Address#AddressLine3),
fieldMapping(Address#AddressType),
fieldMapping(Address#City),
fieldMapping(Address#County),
fieldMapping(Address#Country),
fieldMapping(Address#Description),
fieldMapping(Address#GeocodeStatus),
fieldMapping(Address#PostalCode),
fieldMapping(Address#State),
fieldMapping(Address#ValidUntil),
fieldMapping(Address#AddressLine1Kanji),
fieldMapping(Address#AddressLine2Kanji),
fieldMapping(Address#CityKanji),
fieldMapping(Address#CEDEX),
fieldMapping(Address#CEDEXBureau),
```

Mapping array references of a core application contact

The method `arrayMapping(Entity#ArrayKey)` maps array references for `Contact` entities and subentities. The method by default operates in both directions. You can use the following additional qualifying methods as well:

`withMappingDirection(TO_XML)`

Specifies that the array reference is mapped to `ContactManager`.

`withMappingDirection(TO_BEAN)`

Specifies that the array reference is mapped from `ContactManager`.

For example, in `ClaimCenter`, the array of category scores from a vendor performance review is maintained by `ContactManager`. Therefore, the direction of the array reference is one-way, from `ContactManager`:


```
arrayMapping(Contact#CategoryScores)
    .withMappingDirection(TO_BEAN),
```

withABName(name:String)

Specifies a different name for the array reference on the entity in ContactManager.

Use the `fieldMapping` method to map all the fields of the entity to which an array reference refers.

For example, the array extension example “Extending contacts with an array” on page 171 adds a new entity named `ContactServiceState`. This entity represents a state in which a vendor contact operates. Each vendor can operate in more than one state, so the example adds an array reference to `Contact` named `ContactServiceArea` that references an array of `ContactServiceState` entities.

Note: The example also creates a ContactManager `ContactServiceState` entity and a `ContactServiceArea` array reference on the ContactManager `ABContact` entity.

To transfer this data to and from a core application, you define the array reference and map the fields of the entity to which it refers in `ContactMapper`, as follows:

```
arrayMapping(Contact#ContactServiceArea),
fieldMapping(ContactServiceState#AddressBookUID)
    .withABName(MappingConstants.LINK_ID),
fieldMapping(ContactServiceState#PublicID)
    .withMappingDirection(TO_XML)
    .withABName(MappingConstants.EXTERNAL_PUBLIC_ID),
fieldMapping(ContactServiceState#ServiceState)
```

PolicyCenter mapping of externally specified unique IDs

PolicyCenter specifies unique IDs for the contacts it creates and sends to ContactManager. ContactManager uses these unique IDs for the new contact’s `LinkID` value rather than creating its own unique ID.

The PolicyCenter `ContactMapping` class defines the following mapping for the unique ID of a contact that is to be created in ContactManager:

```
fieldMapping(Contact#ExternalID)
    .withMappingDirection(TO_XML)
    .withABName(EXTERNAL_UNIQUE_ID),
```

See also

- “PolicyCenter contact creation with external unique IDs” on page 196
- “Mapping externally specified unique IDs of a ContactManager contact” on page 277
- “Creating and linking a contact” on page 195
- “ContactManager link IDs and comparison to other IDs” on page 260
- “ABContactAPI methods” on page 264

Special handling in ContactMapper for core applications

In some cases, it is not possible to use `ContactMapper` code to handle the transfer of contact data. In those cases, the core application uses methods that handle more complex cases, like related contacts and addresses.

Special handling in one direction

Address handling is an example of special handling of data sent from ContactManager. Addresses sent from ContactManager require special handling in core applications to support scenarios like swapping a primary address for a secondary address. Turning address data into XML does not require any special handling, so addresses being sent to ContactManager use the normal `ContactMapper` code. For these reasons, the following foreign key and array references are defined as one-way, from the core application to ContactManager:

```
fkMapping(Contact#PrimaryAddress)
    .withMappingDirection(TO_XML),
```

```
arrayMapping(Contact#ContactAddresses)
    .withMappingDirection(TO_XML),
```

When a change to a primary or secondary address for a contact comes in from a core application, the core application calls `populateAddresses` instead of the regular mapping code. The method is defined in `ContactIntegrationXMLMapperAppBase`.

This method call is explicit in `ClaimCenter`:

```
arrayMapping(Contact#ContactAddresses)
    .withMappingDirection(TO_BEAN)
    .withArrayBeanBlock( \ am, bp ->
        populateAddresses(bp.Bean as Contact, bp.XmlBackedInstance)),
```

`PolicyCenter` and `BillingCenter` also call `populateAddresses` for incoming addresses, but the method call is not explicit in `ContactMapper`.

While the address foreign key and address array reference require special handling, the fields of an address do not. You can add or delete address fields as needed by using the `fieldMapping` method.

Special handling in both directions

In `ClaimCenter`, related contacts use a parameter of the `withMappingDirection` method not discussed so far:

```
.withMappingDirection(BOTH)
```

You can use this method to specify one behavior for a field when it is sent to `ContactManager` and another behavior when the field is received from `ContactManager`. For example:

```
fkMapping(ContactContact#RelatedContact)
    .withMappingDirection(BOTH)
    .withABName("RelABContact")
    .withEntityXMLBlock( \ lm, xp -> populateContactXmlForRelatedContact(lm, xp))
    .withEntityBeanBlock( \ lm, bp -> populateBeanFromXml(bp)),
```

Excluding contact fields from ClaimCenter contact synchronization

You can use the `ContactMapper` method `withAffectsSync` to configure fields to be excluded from the set of fields that `ClaimCenter` uses to determine if a contact is synchronized with `ContactManager`. By default, all fields that you add to `ContactMapper` are included in the fields that are checked for synchronization status.

If there is a field in `ContactMapper` that you want excluded from the synchronization check, use the `withAffectsSync` method and set its parameter to `false`. For example, the following code excludes the field `CategoryScores`:

```
arrayMapping(Contact#CategoryScores)
    .withMappingDirection(TO_BEAN)
    .withAffectsSync(false),
```

Note: You do not use `ContactMapper` for fields that determine contact relationships, such as `contactBidiRelCode="employer"`. The inclusion or exclusion of contact relationship fields is defined in the `RelationshipSyncConfig` class.

See also

- “Synchronizing ClaimCenter contact fields” on page 207
- “Synchronizing ClaimCenter and ContactManager contacts” on page 203

Excluding contact fields from being saved in the ClaimCenter database

You can use the `ContactMapper` method `withPersist` to specify that a contact field not be saved in the `ClaimCenter` database. This feature enables `ClaimCenter` to receive the field from `ContactManager` and display it the

ClaimCenter user interface, but not save the field. By default, all contact fields in `ContactMapper` are saved, or *persisted*.

IMPORTANT Setting a field to not persist means that you want the ClaimCenter user to be able to see the value of the field but not change it. Do not enable editing for the field in screens like `ContactDetails.pcf`, and do not send the field back to `ContactManager`.

To set a field to not persist, use the `withPersist` method and set its parameter to `false`. You must also use `withMappingDirection(TO_BEAN)` to map the field only from `ContactManager` to ClaimCenter. Additionally, use `withAffectsSync(false)` to exclude the field from being included in the synchronization check. Otherwise, the contact would always be out of sync. For example:

```
arrayMapping(Contact#ContactManagerOnlyField)
    .withMappingDirection(TO_BEAN)
    .withAffectsSync(false)
    .withPersist(false),
```

Overview of ContactManager plugins

Plugins are software modules that `ContactManager` uses to perform actions or calculate results. Strictly speaking, the term *plugin* refers to an interface, and the term *plugin implementation* is a class that implements a plugin interface. However, in practice, *plugin implementation* is often shortened to just *plugin*.

As described in the *Integration Guide*, you can write your own plugin implementations of Guidewire plugin interfaces in Gosu.

Additionally, there are *plugin registry files*, which you access in `ContactManager` studio. You use a registry to identify which plugin implementation `ContactManager` is to use.

See also

- For a general overview of plugins, see the *Integration Guide*.

Register a plugin implementation in ContactManager

About this task

An example of a plugin registry in `ContactManager` is `PolicySystemPlugin.gwp`, which `ContactManager` can use to send changes in contacts to PolicyCenter.

Procedure

1. Start Guidewire Studio™ for `ContactManager`.

At a command prompt, navigate to the `ContactManager` installation folder and enter the following command:

```
gwb studio
```

2. In the **Project** window, navigate to **configuration→config→Plugins→registry** and then to `PolicySystemPlugin.gwp`, which is a *plugin registry*. Double-click this file to open it in the editor.
3. In the Plugin Registry editor on the right, in the **Class** field, the default plugin that is registered in the base application is `gw.plugin.integration.StandAloneClientSystemPlugin`.

This class is a *plugin implementation* that implements the `ClientSystemPlugin` class. `ContactManager` does not use this plugin implementation to communicate with a core application. Its purpose is to provide a way to demonstrate `ContactManager` without having a core application integrated.

Note: You might have set up `ContactManager` to integrate with PolicyCenter, as described in “Integrating `ContactManager` with PolicyCenter in QuickStart” on page 66. In that case, the class that is registered is `gw.plugin.policy.pc1000.PCPolicySystemPlugin`, which `ContactManager` actually does use to communicate changes to PolicyCenter.

4. If you navigate to **configuration→gsrc** and then to `gw.plugin.policy.pc1000.PCPolicySystemPlugin`, you can see that this plugin implementation extends the `ClientSystemPlugin1000` class. The `ClientSystemPlugin1000` class extends the `AbstractClientSystemPlugin` class, which implements `ClientSystemPlugin`.

ContactManager plugins

The following table lists most of the ContactManager plugins, except for the messaging plugins.

IMPORTANT If you want to change how ContactManager communicates with a core application to send contact updates with `BillingSystemPlugin`, `ClaimSystemPlugin`, or `PolicySystemPlugin`, extend `gw.plugin.AbstractClientSystemPlugin`. Do not implement the plugin interface `gw.plugin.ClientSystemPlugin`. Implementing the `ClientSystemPlugin` interface will prevent ContactManager from starting.

Registry	Description
AuthenticationSourceCreatorPlugin	<ul style="list-style-type: none"> Registry – <code>configuration→config→Plugins→registry→AuthenticationSourceCreatorPlugin.gwp</code> Plugin Interface – <code>gw.plugin.security.AuthenticationSourceCreatorPlugin</code> Default Registered Plugin Implementation – <code>com.guidewire.pl.system.security.impl.DefaultAuthenticationServicePlugin</code> <p>Authenticates the user logging in to ContactManager.</p>
BillingSystemPlugin	<ul style="list-style-type: none"> Registry – <code>configuration→config→Plugins→registry→BillingSystemPlugin.gwp</code> Plugin Interface – <code>gw.plugin.ClientSystemPlugin</code> Note: Do not implement this interface. Instead, extend <code>gw.plugin.AbstractClientSystemPlugin</code>. Default registered sample plugin implementation – <code>gw.plugin.integration.StandAloneClientSystemPlugin</code> Working integration plugin class implementation – <code>gw.plugin.billing.cc1000.BCBillingSystemPlugin</code> Parent class of working integration plugin – <code>gw.plugin.integration.ClientSystemPlugin1000</code> <p>Sends changes in contacts to BillingCenter. See “Integrating ContactManager with Guidewire core applications” on page 59.</p>
ClaimSystemPlugin	<ul style="list-style-type: none"> Registry – <code>configuration→config→Plugins→registry→ClaimSystemPlugin.gwp</code> Plugin Interface – <code>gw.plugin.ClientSystemPlugin</code> Note: Do not implement this interface. Instead, extend <code>gw.plugin.AbstractClientSystemPlugin</code>. Default registered sample plugin implementation – <code>gw.plugin.integration.StandAloneClientSystemPlugin</code> Working integration plugin class implementation – <code>gw.plugin.claim.cc1000.CCClaimSystemPlugin</code> Parent class of working integration plugin – <code>gw.plugin.integration.ClientSystemPlugin1000</code> <p>Sends changes in contacts to ClaimCenter. See “Integrating ContactManager with Guidewire core applications” on page 59.</p>
GeocodePlugin	<ul style="list-style-type: none"> Registry – <code>configuration→config→Plugins→registry→GeocodePlugin.gwp</code> Parent Class – <code>gw.api.geocode.AbstractGeocodePlugin</code> Plugin Class Implementation – <code>gw.plugin.geocode.impl.BingMapsPluginRest</code>

Registry	Description
	Connects with a geocoding service to provide geocoding information for addresses. For information on setting up geocoding with this plugin, see “Geocoding and proximity search for vendor contacts” on page 108.
IABContactScoringPlugin	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→IABContactScoringPlugin.gwp Plugin Interface – <code>gw.plugin.contact.IABContactScoringPlugin</code> Plugin Class Implementation – <code>gw.plugin.spm.impl.ABContactScoringPlugin</code> <p>Scores provider reviews sent from ClaimCenter. .</p>
IAddressAutocompletePlugin	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→IAddressAutocompletePlugin.gwp Plugin Interface – <code>gw.plugin.addressautocomplete.IAddressAutocompletePlugin</code> Plugin Class Implementation – <code>gw.api.address.DefaultAddressAutocompletePlugin</code> <p>Use this plugin to configure how automatic address completion and fill-in operate. See the <i>Globalization Guide</i>.</p>
IEncryption	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→IEncryption.gwp Plugin Interface – <code>gw.plugin.util.IEncryption</code> Plugin Class Implementation – <code>gw.plugin.encryption.EncryptionByReversePlugin</code> <p>Encodes or decodes a String based on an algorithm you provide to hide important data, such as bank account numbers or private personal data. ContactManager does not provide any encryption algorithm in the product. ContactManager simply calls the EncryptionByReversePlugin implementation, which does nothing. See the <i>Integration Guide</i> for information on encryption integration.</p>
IFindDuplicatesPlugin	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→IFindDuplicatesPlugin.gwp Plugin Interface – <code>gw.plugin.contact.IFindDuplicatesPlugin</code> Plugin Class Implementation – <code>gw.plugin.contact.findduplicates.FindDuplicatesPlugin</code> <p>Finds duplicate contacts. See “IFindDuplicatesPlugin plugin interface” on page 289.</p>
IGroupExceptionPlugin	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→IGroupExceptionPlugin.gwp Plugin Interface – <code>gw.plugin.exception.IGroupExceptionPlugin</code> Plugin Class Implementation – <code>com.guidewire.pl.domain.escalation.RulesBasedGroupExceptionPlugin</code> <p>Calls the group exception rule set. See the <i>Integration Guide</i> for information on exception and escalation plugins.</p>
IPhoneNormalizerPlugin	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→IPhoneNormalizerPlugin.gwp Plugin Interface – <code>gw.plugin.phone.IPhoneNormalizerPlugin</code> Parent Class of Registered Plugin – <code>gw.api.phone.DefaultPhoneNormalizerPlugin</code> Default Registered Plugin Class – <code>gw.plugin.phone.DefaultABPhoneNormalizerPlugin</code> <p>The default registered plugin class extends the <code>DefaultPhoneNormalizerPlugin</code> class, which extends <code>AbstractPhoneNormalizerPlugin</code>. <code>DefaultABPhoneNormalizerPlugin</code> provides support for phone numbers to <code>ABContact</code> and its subtypes.</p>

Registry	Description
ITestingClock	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→ITestingClock.gwp Plugin Interface – gw.plugin.system.ITestingClock Parent Class of Registered Plugin – com.guidewire.pl.plugin.system.internal.LongBasedTestingClock Plugin Class Implementation – com.guidewire.pl.plugin.system.internal.OffsetTestingClock <p>The default registered plugin class extends the LongBasedTestingClock class, which extends com.guidewire.pl.plugin.system.internal.ClusterWideTestingClockBase<java.lang.Long>. Used for testing complex behavior over a long span of time, such as timeouts that are multiple days or weeks later. This plugin is for development (non-production) use only. It programmatically changes the system time to simulate passage of time in ContactManager.</p> <p>IMPORTANT You must never use the testing clock plugin on a production server. See “Testing clock plugin interface” on page 299.</p>
IUserExceptionPlugin	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→IUserExceptionPlugin.gwp Plugin Interface – gw.plugin.exception.IUserExceptionPlugin Plugin Class Implementation – com.guidewire.pl.domain.escalation.RulesBasedUserExceptionPlugin <p>Calls the user exception rule set. See the <i>Integration Guide</i> for information on exception and escalation plugins.</p>
OfficialIdToTaxIdMappingPlugin	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→OfficialIdToTaxIdMappingPlugin.gwp Plugin Interface – gw.plugin.contact.OfficialIdToTaxIdMappingPlugin Plugin Class Implementation in ContactManager, ClaimCenter, and BillingCenter – com.guidewire.pl.plugin.contact.internal.AlwaysFalseOfficialIdToTaxIdMappingPlugin Plugin Class Implementation in PolicyCenter – gw.plugin.contact.impl.PCOfficialIdToTaxIdMappingPlugin <p>In the base configuration, this plugin is used only by PolicyCenter. However, it is available to all core applications. In the base configurations of ContactManager, ClaimCenter, and BillingCenter, the registered plugin is AlwaysFalseOfficialIDToTaxIDMappingPlugin, which always returns false. ContactManager, ClaimCenter, and BillingCenter support only a single TaxID field and do not need to do this mapping.</p> <p>In the base configuration of PolicyCenter, the registered plugin is gw.plugin.contact.impl.PCOfficialIdToTaxIdMappingPlugin. PolicyCenter supports an array of official IDs for each contact. For the integration with ContactManager to work, PolicyCenter needs to be able to map one of the typecodes from its OfficialIDType typelist to the contact's TaxID field. In PolicyCenter, this field is the contact's primary ID for identification purposes.</p> <p>In the base configuration of PolicyCenter, the plugin implementation overrides the method isTaxId. This method override determines if the tax ID is in the OfficialIDType typelist as a Social Security Number (TC_SSN) or a Federal Employer Identification Number (TC_FEIN). You can extend PolicyCenter with your own official IDs and map the ones that are appropriate for your locale or country.</p>

Registry	Description
PendingContactChangeConfigurationPlugin	<ul style="list-style-type: none"> Registry – <code>configuration→config→Plugins→registry→PendingContactChangeConfigurationPlugin.gwp</code> Plugin Interface – <code>gw.plugin.contact.PendingContactChangeConfigurationPlugin</code> Plugin Class Implementation – <code>gw.plugin.contact.pendingchange.PendingContactChangeConfigurationPluginImpl</code> <p>Controls the matching used in generating the contact difference view in the Pending Updates screen. To configure how pending updates work, write a plugin that implements the interface.</p>
PolicySystemPlugin	<ul style="list-style-type: none"> Registry – <code>configuration→config→Plugins→registry→PolicySystemPlugin.gwp</code> Plugin Interface – <code>gw.plugin.ClientSystemPlugin</code> Note: Do not implement this interface. Instead, extend <code>gw.plugin.AbstractClientSystemPlugin</code>. Default registered plugin implementation – <code>gw.plugin.integration.StandAloneClientSystemPlugin</code> Working integration plugin class implementation – <code>gw.plugin.policy.cc1000.PCPolicySystemPlugin</code> Parent class of working integration plugin – <code>gw.plugin.integration.ClientSystemPlugin1000</code> <p>Sends changes in contacts to PolicyCenter. See “Integrating ContactManager with Guidewire core applications” on page 59.</p>
ValidateABContactCreationPlugin	<ul style="list-style-type: none"> Registry – <code>configuration→config→Plugins→registry→ValidateABContactCreationPlugin.gwp</code> Plugin Interface – <code>gw.plugin.contact.ValidateABContactCreationPlugin</code> Parent Class of Registered Plugin Class – <code>gw.plugin.contact.ValidateABContactCreationPluginBase</code> Default Registered Plugin Class – <code>gw.plugin.contact.ValidateABContactCreationPluginImpl</code> <p>Validates that creation criteria have been met before creating a contact. See “ValidateABContactCreationPlugin interface” on page 297.</p>
ValidateABContactSearchCriteriaPlugin	<ul style="list-style-type: none"> Registry – <code>configuration→config→Plugins→registry→ValidateABContactSearchCriteriaPlugin.gwp</code> Plugin Interface – <code>gw.plugin.contact.ValidateABContactSearchCriteriaPlugin</code> Parent Class of Registered Plugin Class – <code>gw.plugin.contact.ValidateABContactSearchCriteriaPluginBase</code> Default Registered Plugin Class – <code>gw.plugin.contact.ValidateABContactSearchCriteriaPluginImpl</code> <p>Validates that search criteria have been met for finding a contact. To specify that a Contact entity field is required in a search, add it to <code>gw.plugin.contact.ValidateABContactSearchCriteriaPluginImpl</code>. The following code snippet from <code>ValidateABContactSearchCriteriaPluginBase</code> shows the default search criteria defined for an ABPerson entity:</p>

```
protected function abPersonCanSearch(
    searchCriteria : ABContactSearchCriteria) : boolean {
    if (searchCriteria.FirstName != null or
        searchCriteria.FirstNameKanji != null) {
```

Registry	Description
	<pre> if (searchCriteria.Keyword == null and searchCriteria.KeywordKanji == null) { return false } } if (searchCriteria.Keyword == null and searchCriteria.KeywordKanji == null and searchCriteria.FirstName == null and searchCriteria.FirstNameKanji == null and searchCriteria.TaxID == null and satisfiesNoLocaleSpecificCriteriaRequirements(searchCriteria) and searchCriteria.Address.PostalCode == null and not searchCriteria.isValidProximitySearch()){ return false } return true </pre>
WebservicesAuthenticationPlugin	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→WebservicesAuthenticationPlugin.gwp Plugin Interface – gw.plugin.security.WebservicesAuthenticationPlugin Plugin Class Implementation – gw.plugin.security.DefaultWebservicesAuthenticationPlugin <p>For WS-I web services only, configures custom authentication logic. See the <i>Integration Guide</i> for information on the web services authentication plugin.</p>

The following table lists the messaging plugins.

See also the *Integration Guide* for information on messaging and events.

Registry	Description
BCBillingSystemTransport	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→BCBillingSystemTransport.gwp Interfaces – <ul style="list-style-type: none"> gw.plugin.messaging.MessageTransport gw.plugin.InitializablePlugin Plugin Class Implementation – gw.plugin.integration.InitializableClientSystemTransport
CCClaimSystemTransport	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→CCClaimSystemTransport.gwp Interfaces – <ul style="list-style-type: none"> gw.plugin.messaging.MessageTransport gw.plugin.InitializablePlugin Plugin Class Implementation – gw.plugin.integration.InitializableClientSystemTransport
PCPolicySystemTransport	<ul style="list-style-type: none"> Registry – configuration→config→Plugins→registry→PCPolicySystemTransport.gwp Interfaces – <ul style="list-style-type: none"> gw.plugin.messaging.MessageTransport gw.plugin.InitializablePlugin Plugin Class Implementation – gw.plugin.integration.InitializableClientSystemTransport

Registry	Description
documentStoreTransport	<ul style="list-style-type: none"> Registry – <code>configuration→config→Plugins→registry→documentStoreTransport.gwp</code> Interfaces – <ul style="list-style-type: none"> <code>gw.plugin.messaging.MessageTransport</code> <code>gw.plugin.InitializablePlugin</code> Plugin Class Implementation – <code>gw.plugin.document.impl.DocumentStoreTransport</code>
emailMessageTransport	<ul style="list-style-type: none"> Registry – <code>configuration→config→Plugins→registry→emailMessageTransport.gwp</code> Parent Class – <code>gw.api.email.AbstractEmailMessageTransport</code> Plugin Class Implementation – <code>gw.plugin.email.impl.EmailMessageTransport</code>

IFindDuplicatesPlugin plugin interface

The FindDuplicatesPlugin plugin implementation implements the IFindDuplicatesPlugin plugin interface in Gosu and is available in Guidewire Studio™ for ContactManager. To view or edit the plugin implementation, start ContactManager Studio. Then, in the **Project** window, navigate to `configuration→gsrc` and then to `gw.plugin.contact.findduplicates.FindDuplicatesPlugin`.

Note: The FindDuplicatesPlugin plugin implementation and its helper classes use *Gosu generics*.

For more information, see the *Gosu Reference Guide*.

The plugin attempts to find exact and potential matches for the subtype of ABContact that is passed to it. If no duplicate finder exists for the subtype, the plugin traverses the ABContact tree to find the closest parent type it can use in the search.

The plugin uses a set of duplicate finder classes, which define potential and exact match logic for a set of ABContact subtypes. The plugin sets up the duplicate finder classes in two Map definitions:

```
static final var WIDE_MAP : Map<typekey.ABContact,
    Type<DuplicateFinderBase>> = {
    typekey.ABContact.TC_ABCOMPANY ->
        CompanyDuplicateFinder<ABCompany>,
    typekey.ABContact.TC_ABPERSO ->
        PersonDuplicateFinder<ABPerson>,
    typekey.ABContact.TC_ABPLACE ->
        PlaceDuplicateFinder
}
static final var DEFAULT_MAP : Map<typekey.ABContact,
    Type<DuplicateFinderBase>> = {
    typekey.ABContact.TC_ABCOMPANYVENDOR ->
        CompanyDuplicateFinder<ABCompanyVendor>,
    typekey.ABContact.TC_ABCOMPANY ->
        CompanyDuplicateFinder<ABCompany>,
    typekey.ABContact.TC_ABPERSO ->
        PersonVendorDuplicateFinder,
    typekey.ABContact.TC_ABUSERCONTACT ->
        UserDuplicateFinder,
    typekey.ABContact.TC_ABPERSO ->
        PersonDuplicateFinder<ABPerson>,
    typekey.ABContact.TC_ABPLACE ->
        PlaceDuplicateFinder
}
```

These two Map definitions are likely to be the only area of the plugin itself that you need to edit. For example, if you add a duplicate finder class for the ABAttorney subtype, you could add a typekey definition for that class to the DEFAULT_MAP definition. You do not have to add a map definition for any subtype, even a subtype extension that you create. Because the plugin traverses the ABContact tree to find the nearest parent type, any subtype of ABContact already has matching logic defined for it. Add a duplicate finder class only if you need special matching logic for a subtype.

Note: In the base configuration, `WIDE_MAP` is configured with fewer subtypes than `DEFAULT_MAP`, resulting in looser matching. This behavior is the intended use of these two variables. However, there is no semantic difference between `WIDE_MAP` and `DEFAULT_MAP`. For example, if you added more subtypes to `WIDE_MAP`, you could make search results using `WIDE_MAP` more specific than `DEFAULT_MAP`.

The `WIDE_MAP` definition is used by the ContactManager batch process Duplicate Contact Finder in the configuration parameter `DuplicateContactWideSearch`. Additionally, it is used in contact linking calls from ClaimCenter.

See also

- “Changing match results and search scope settings” on page 251
- “Detecting and merging duplicate contacts” on page 247
- “Linking in ClaimCenter” on page 197

Duplicate finder classes

`FindDuplicatesPlugin` uses a set of duplicate finder classes, each of which defines:

- The minimum set of fields required to attempt a match.
- The fields required for exact matches for the `ABContact` subtype.
- The fields required for potential matches for the `ABContact` subtype.

Each class checks for the existence of a minimum set of fields for performing a match with this subtype. If there is enough data for a match, the class can perform a search for potential matches and check the potential matches to see if any are an exact match.

If there is not enough data for a match, the `validateMandatoryFields` method throws a `TooLooseContactDuplicateMatchCriteriaException`. These exceptions use display keys, which you can access in Guidewire Studio™ for ContactManager. Navigate in the **Project** window to **configuration**→**config**→**Localizations**→**Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor. In the editor, press `Ctrl+F` and search for `TooLooseContactDuplicateMatchCriteriaException`. To view or edit the duplicate finder classes, start ContactManager Studio. Then, in the **Project** window, navigate to **configuration**→**gsrc** and then to `gw.plugin.contact.findduplicates`. Under this node, you see all the duplicate finder classes.

In these classes, you can change the logic required for matching a subtype. For example, you want to add a new field to the potential match logic in one of the duplicate finder classes. You must first add appropriate matching code for the field to the query builder class that supports that duplicate finder class. You then add the query finder method call to the duplicate finder class. For an example of a class that adds field matching logic, see “`PersonDuplicateFinder` class” on page 291.

The query builder classes are located one node down in the **Project** window. Navigate to **configuration**→**gsrc** and then to `gw.plugin.contact.findduplicates.querybuilder`. For more information on these classes, see “Query builder classes” on page 293.

You also might want to define new matching for an `ABContact` subtype that does not have a duplicate finder class. In this case, you would create a new duplicate finder class and a query builder class for the subtype.

Note: You do not have to add a duplicate finder class for any subtype, even a subtype extension that you create. Because the plugin traverses the `ABContact` tree to find the nearest parent type, any subtype of `ABContact` already has matching logic defined for it. Add a duplicate finder class only if you need special matching logic for a subtype.

The duplicate finder class descriptions that follow describe the functionality available in the base product. The `CompanyDuplicateFinder` description includes code for required fields, potential match, and exact match. The code in the other duplicate finders is similar.

CompanyDuplicateFinder class

This Gosu class defines duplicate contact matching for the `ABCompany` subtype. If the subtype is `ABCompanyVendor`, `ABAutoRepairShop`, `ABAutoTowingAgcy`, `ABLawFirm`, or `ABMedicalCareOrg`, matching is performed for `ABCompany`. The class uses the query builder `CompanyQueryBuilder`, described at “`CompanyQueryBuilder` class” on page 295.

To open this class in Guidewire Studio™, navigate in the **Project** window to **configuration→gsrsrc** and then to `gw.plugin.contact.findduplicates.CompanyDuplicateFinder`.

Note: The match criteria descriptions that follow includes the code for required fields, potential match, and exact match. The code in the other duplicate finders is similar.

`CompanyDuplicateFinder` defines the following match criteria.

- Fields to match:
 - Name – Match this field as starts with or equals, depending on context.
 - PrimaryAddress (AddressLine1, City, State, PostalCode) – Match these PrimaryAddress fields as equals.
 - TaxID – Match this field as equals.
 - WorkPhone or FaxPhone – Match any single phone field as equals.
- Required fields: Name and at least one of PrimaryAddress, TaxID, or any phone field: WorkPhone or FaxPhone.

The following code defines the required fields for an `ABCompany`:

```
override function validateMandatoryFields() {
    if (_searchContact.Name == null or
        (hasNoPrimaryAddress() and
         hasNoPhoneNumber() and
         _searchContact.TaxID == null))
        throwException(_searchContact)
}
```

- Potential match types:
 - Starts with Name and equals either PrimaryAddress or a phone field.
 - Equals TaxID.

The following code defines the potential match fields for an `ABCompany`:

```
override function makeQueries() : List<Query<C>> {
    var queries = new ArrayList<Query<C>>()
    //Query: TaxID
    new CompanyQueryBuilder<C>(_searchContact)
        .hasEqualTaxId()//AND
        .buildAndAdd(queries)
    //Query: Name and PhoneNumber
    if (not hasNoPhoneNumber()) {
        new CompanyQueryBuilder<C>(_searchContact)
            .startsWithName()//AND
            .hasEqualPhoneNumbers()
            .buildAndAdd(queries)
    }
    //Query: Name and Address
    if (not hasNoPrimaryAddress()) {
        new CompanyQueryBuilder<C>(_searchContact)
            .startsWithName()//AND
            .hasEqualAddress()
            .buildAndAdd(queries)
    }
    return queries
}
```

- Exact Match – Equals both TaxID and Name.

The following code defines the exact match fields for an `ABCompany`:

```
override function isExactMatch(
    searchContact : C, resultABContact : C) : boolean {
    return equalsAndNotNull<String>(
        searchContact.TaxID, resultABContact.TaxID) &&
        equalsAndNotNull<String>(
            searchContact.Name, resultABContact.Name)
}
```

PersonDuplicateFinder class

This Gosu class defines matching for the `ABPerson` subtype. If the subtype is `ABAdjudicator`, matching is performed for `ABPerson`.

To open this class in Guidewire Studio™, navigate in the **Project** window to **configuration→gsrsrc** and then to `gw.plugin.contact.findduplicates.PersonDuplicateFinder`.

The class uses the query builder `PersonQueryBuilder`, described at “`PersonQueryBuilder` class” on page 296. That query builder extends `PersonQueryBuilderBase`, which adds logic for matching the first name, last name, date of birth, and phone numbers of the contacts. You can use that class and this one as an example of how to add matching logic for a field to a query builder and a Duplicate Finder.

`PersonDuplicateFinder` defines the following match criteria:

- Fields to match:
 - `FirstName` – Match this field as *starts with* or *equals*, depending on context.
 - `LastName` – Match this field as *equals*.
 - `PrimaryAddress` (`AddressLine1`, `City`, `State`, `PostalCode`) – Match these `PrimaryAddress` fields as *equals*.
 - `LicenseNumber` and `LicenseState` – Match both Driver’s License fields as *equals*.
 - `DateOfBirth` – Match this field as *equals*.
 - `TaxID` – Match this field as *equals*.
 - `HomePhone`, `WorkPhone`, `FaxPhone`, or `CellPhone` – Match any single phone field as *equals*.
- Required fields: `FirstName` and `LastName` and at least one of `PrimaryAddress`, `DateOfBirth`, `TaxID`, any phone field, or `LicenseNumber` and `LicenseState`.
- Match types:
 - Potential – Starts with `FirstName` and *equals* `LastName` and *equals* one of `PrimaryAddress`, `DateOfBirth`, any phone field, or `LicenseNumber` and `LicenseState`.
 - Potential – *Equals* `LastName` and `TaxID`.
 - Exact – *Equals* `FirstName` and `LastName` and `DateOfBirth`, and *equals* one of `PrimaryAddress` or any phone field.
 - Exact – *Equals* `FirstName` and `LastName` and `TaxID`.

PersonVendorDuplicateFinder class

This Gosu class defines matching for the `ABPersonVendor` subtype. If the subtype is `ABAttorney` or `ABDoctor`, matching is performed for `ABPersonVendor`.

To open this class in Guidewire Studio™, navigate in the **Project** window to **configuration→gsrsrc** and then to `gw.plugin.contact.findduplicates.PersonVendorDuplicateFinder`.

The class uses the query builder `PersonQueryBuilder`, described at “`PersonQueryBuilder` class” on page 296.

`PersonDuplicateFinder` defines the following match criteria:

- Fields to match:
 - `FirstName` – Match this field as *starts with* or *equals*, depending on context.
 - `LastName` – Match this field as *equals*.
 - `PrimaryAddress` (`AddressLine1`, `City`, `State`, `PostalCode`) – Match these `PrimaryAddress` fields as *equals*.
 - `TaxID` – Match this field as *equals*.
 - `HomePhone`, `WorkPhone`, `FaxPhone`, or `CellPhone` – Match any single phone field as *equals*.
- Required fields: `FirstName` and `LastName` and at least one of `PrimaryAddress`, `TaxID`, or any phone field.
- Match types:
 - Potential – Starts with `FirstName` and *equals* `LastName` and *equals* one of `PrimaryAddress` or any phone field.
 - Potential – *Equals* `TaxID`.
 - Exact – *Equals* `FirstName` and `LastName` and `TaxID`.

PlaceDuplicateFinder class

This Gosu class defines matching for the `ABPlace` subtype. If the subtype is `ABLegalVenue`, matching is performed for `ABPlace`.

To open this class in Guidewire Studio™, navigate in the **Project** window to **configuration→gsrc** and then to `gw.plugin.contact.findduplicates.PlaceDuplicateFinder`.

The class uses the query builder `PlaceQueryBuilder`, described at “`PlaceQueryBuilder` class” on page 295.

`PlaceDuplicateFinder` defines the following match criteria:

- Fields to match:
 - `Name` – Match this field as *starts with* or *equals*, depending on context.
 - `PrimaryAddress` (`AddressLine1`, `City`, `State`, `PostalCode`) – Match these `PrimaryAddress` fields as *equals*.
- Required fields: `Name` and `PrimaryAddress`.
- Match types:
 - Potential – Starts with `Name`.
 - Potential – Equals `PrimaryAddress`.
 - Exact – Equals `Name` and `AddressLine1` and `City` and `State` and `PostalCode`.

UserDuplicateFinder class

This class defines matching for the `ABUserContact` subtype.

To open this class in Guidewire Studio™, navigate in the **Project** window to **configuration→gsrc** and then to `gw.plugin.contact.findduplicates.UserDuplicateFinder`.

The class uses the query builder `UserContactQueryBuilder`, described at “`UserContactQueryBuilder` class” on page 296.

`UserContactDuplicateFinder` defines the following match criteria:

- Fields to match:
 - `FirstName` – Match this field as *starts with* or *equals*, depending on context.
 - `LastName` – Match this field as *equals*.
 - `EmployeeNumber` – Match this field as *equals*.
- Required fields: `FirstName` and `LastName` and at least one of `PrimaryAddress`, `TaxID`, or any phone field.
- Match types:
 - Potential – Starts with `FirstName` and equals `LastName`.
 - Potential – Equals `EmployeeNumber`.
 - Exact – Equals `EmployeeNumber`.

Query builder classes

The query builder Gosu classes build queries that the duplicate finder classes use to search the database for specific subtypes of `ABContact`. They define queries that:

- Compare fields to see if they are equal. See `hasEqualTaxID` under “`ContactQueryBuilder` class” on page 293.
- Compare fields to see if they start with the same characters. See `startsWithName` under “`ContactQueryBuilder` class” on page 293.
- Compare a set of fields to see if they are all equal. See `hasEqualAddress` under “`ContactQueryBuilder` class” on page 293.
- Compare a set of fields to see if one of them is equal. See `hasEqualPhoneNumbers` under “`ContactQueryBuilder` class” on page 293.

For information on the duplicate finder classes, see “Duplicate finder classes” on page 290.

ContactQueryBuilder class

This Gosu class provides a basic set of `ABContact` queries for the query builder classes. The classes that extend this one produce a set of queries for specific `ABContact` subtypes, which in turn are used by the duplicate finder classes for those subtypes.

To open this class in Guidewire Studio™, navigate in the **Project** window to **configuration→gsrc** and then to `gw.plugin.contact.findduplicates.querybuilder.ContactQueryBuilder`.

The class builds the following queries:

hasEqualTaxId

Adds an expression to the query. The expression determines if the TaxID field of the ABContact being checked for duplicates is equal to the TaxID field of an ABContact in the database.

```
function hasEqualTaxId() : U {
    addExpression(new EqualFieldExpression<T>{
        "TaxID", _contact.TaxID})
    return this as U
}
```

hasEqualPhoneNumbers

Adds an expression to the query. Determines if the HomePhone, WorkPhone, or FaxPhone field of the ABContact being checked for duplicates is equal to the equivalent field of an ABContact in the database. If one of the fields matches, the contacts have equal phone numbers.

```
function hasEqualPhoneNumbers() : U {
    var numbers = PhoneNumbers
    var phoneOperators : List<FieldExpression<T>> = {
        new InFieldExpression<T>("HomePhone", numbers),
        new InFieldExpression<T>("WorkPhone", numbers),
        new InFieldExpression<T>("FaxPhone", numbers)
    }
    addExpression(new OrCompositeFieldExpression<T>{
        phoneOperators.toArray() })
    return this as U
}
```

hasEqualAddress

Adds an expression to the query. Compares fields of the PrimaryAddress of the ABContact being checked for duplicates to the equivalent fields of the PrimaryAddress of an ABContact in the database. If the AddressLine1, State, City, and PostalCode fields are equal for both contacts, the addresses are considered equal.

```
function hasEqualAddress() : U {
    addExpression(new AndCompositeFieldExpression<T>({
        new EqualFieldExpression<T>{
            "AddressLine1", "PrimaryAddress",
            _contact.PrimaryAddress.AddressLine1,false),
        new EqualFieldExpression<T>{
            "State", "PrimaryAddress",
            _contact.PrimaryAddress.State,false),
        new EqualFieldExpression<T>{
            "City", "PrimaryAddress",
            _contact.PrimaryAddress.City,false),
        new EqualFieldExpression<T>{
            "PostalCode", "PrimaryAddress",
            _contact.PrimaryAddress.PostalCode,false)
    })))
    return this as U
}
```

startsWithName

Adds an expression to the query. Uses the entire string in the Name field of the ABContact being checked for duplicates. Compares this string to a substring at the start of the Name field of an ABContact in the database. If the string and the substring are equal, the contacts are a starts with match. For example, “Asim” would be a starts with match with “Asimov”.

```
function startsWithName() : U {
    addExpression(new StartsWithFieldExpression<T>("Name", _contact.Name))
    return this as U
}
```

CompanyQueryBuilder class

This Gosu class builds queries for an ABCompany entity. It extends `ContactQueryBuilder`, making the query logic defined in that class available to the class `CompanyDuplicateFinder`.

To open this class in Guidewire Studio™, navigate in the **Project** window to **configuration→gsrc** and then to `gw.plugin.contact.findduplicates.querybuilder.CompanyQueryBuilder`.

See also

- “ContactQueryBuilder class” on page 293
- “CompanyDuplicateFinder class” on page 290

PlaceQueryBuilder class

This Gosu class builds queries for an ABPlace entity. It extends `ContactQueryBuilder`, making the query logic defined in that class available to the class `PlaceDuplicateFinder`.

To open this class in Guidewire Studio™, navigate in the **Project** window to **configuration→gsrc** and then to `gw.plugin.contact.findduplicates.querybuilder.PlaceQueryBuilder`.

See also

- “ContactQueryBuilder class” on page 293
- “PlaceDuplicateFinder class” on page 292

PersonQueryBuilderBase class

This Gosu class builds queries for querying an ABPerson entity. It extends `ContactQueryBuilder` and adds new query logic used by `PersonQueryBuilder` and `UserContactQueryBuilder`. You can compare `PersonQueryBuilderBase` to “ContactQueryBuilder class” on page 293 to see how to add new query logic for a field, such as `LastName` or `LicenseNumber`.

To open this class in Guidewire Studio™, navigate in the **Project** window to **configuration→gsrc** and then to `gw.plugin.contact.findduplicates.querybuilder.PersonQueryBuilderBase`.

The class builds the following queries in addition to, or in place of, those in `ContactQueryBuilder`:

startsWithFirstName

Adds an expression to the query. Uses the entire string in the `FirstName` field of the ABPerson being checked for duplicates. Compares this string to a substring at the start of the `FirstName` field of an ABPerson in the database. If the string and the substring are equal, the contacts are a *starts with* match. For example, “Asim” would be a starts with match with “Asimov”.

```
function startsWithFirstName() : U {  
    addExpression(new StartsWithFieldExpression<T>(  
        "FirstName", Contact.FirstName))  
    return this as U  
}
```

hasEqualLastName

Adds an expression to the query. The expression determines if the `LastName` field of the ABPerson being checked for duplicates is equal to the `LastName` field of an ABPerson in the database.

```
function hasEqualLastName() : U {  
    addExpression(new EqualFieldExpression<T>(  
        "LastNameDenorm", Contact.LastName, false))  
    return this as U  
}
```

hasEqualBirthDate

Adds an expression to the query. The expression determines if the `DateOfBirth` field of the ABPerson being checked for duplicates is equal to the `DateOfBirth` field of an ABPerson in the database.


```
function hasEqualBirthDate() : U {
    addExpression(new EqualFieldExpression<T>({
        "DateOfBirth", Contact.DateOfBirth))
    return this as U
}
```

hasEqualLicenseNumber

Adds an expression to the query. The expression determines if the `LicenseNumber` and `LicenseState` fields of the `ABPerson` being checked for duplicates are equal to the same fields of an `ABPerson` in the database.

```
function hasEqualLicenseNumber() : U {
    addExpression(new AndCompositeFieldExpression<T>({
        new EqualFieldExpression<T>("LicenseNumber", Contact.LicenseNumber),
        new EqualFieldExpression<T>("LicenseState", Contact.LicenseState)
    }))
    return this as U
}
```

hasEqualPhoneNumbers

Adds an expression to the query. The method overrides `ContactQueryBuilder.hasEqualPhoneNumbers` and adds the `CellPhone` field as one of the phone number fields to check. The method determines if the `HomePhone`, `WorkPhone`, `FaxPhone`, or `CellPhone` field of the `ABPerson` being checked for duplicates is equal to the equivalent field of an `ABPerson` in the database. If one of the fields matches, the contacts have equal phone numbers.

```
override function hasEqualPhoneNumbers() : U {
    var numbers = PhoneNumbers
    var phoneOperators : List<FieldExpression<T>> = {
        new InFieldExpression<T>("HomePhone", numbers),
        new InFieldExpression<T>("WorkPhone", numbers),
        new InFieldExpression<T>("FaxPhone", numbers),
        new InFieldExpression<T>("CellPhone", numbers)
    }
    addExpression(new OrCompositeFieldExpression<T>({
        phoneOperators.toArray()
    }))
    return this as U
}
```

PersonQueryBuilder class

This Gosu class builds queries for an `ABPerson` entity. It extends `PersonQueryBuilderBase`, making the query logic defined in that class available to the classes `PersonDuplicateFinder` and `PersonVendorDuplicateFinder`.

To open this class in Guidewire Studio™, navigate in the **Project** window to **configuration→gsrsrc** and then to `gw.plugin.contact.findduplicates.querybuilder.PersonQueryBuilder`.

See also

- “`PersonQueryBuilderBase` class” on page 295
- “`PersonDuplicateFinder` class” on page 291
- “`PersonVendorDuplicateFinder` class” on page 292

UserContactQueryBuilder class

This Gosu class builds queries for an `ABUserContact` entity. It extends `PersonQueryBuilderBase`, making the query logic defined in that class available to the class `UserDuplicateFinder`. In addition, it adds a method, `hasEqualEmployeeNumber` for comparing equality of the `EmployeeNumber` fields of the two contacts.

To open this class in Guidewire Studio™, navigate in the **Project** window to **configuration→gsrsrc** and then to `gw.plugin.contact.findduplicates.querybuilder.UserContactQueryBuilder`.

See also

- “PersonQueryBuilderBase class” on page 295
- “UserDuplicateFinder class” on page 293

ValidateABContactCreationPlugin plugin interface

The registry file for this plugin is `ValidateABContactCreationPlugin.gwp`. This plugin interface is implemented in Gosu in the abstract class `ValidateABContactCreationPluginBase`. The class `ValidateABContactCreationPluginImpl` extends this base class and is the default plugin class registered in the base configuration.

The class `ValidateABContactCreationPluginBase` implements most of the logic that validates new `ABContact` instances and instances of `ABContact` subentities.

To extend the contact creation logic, such as for a new subentity, open Guidewire Studio™. Then navigate in the Studio **Project** window to **configuration**→**gsrsrc** and then to `gw.plugin.contact.ValidateABContactCreationPluginImpl`.

Contact minimum creation requirements

The `ValidateABContactCreationPluginImpl` class enables you to provide your own definition of the minimum criteria required for creating new contacts. In the base configuration, all contacts are required to have a contact tag. In the base configuration, `ValidateABContactCreationPluginBase` implements the following minimum creation requirements:

- **ABPerson** – Last name, contact tag, and one of the following:
 - Primary address – Address line 1, city, state, ZIP code
 - Phone number – Any of the phone numbers, such as home, cell, fax, or work phone
 - Tax ID
 - Date of birth
 - Driver’s license and driver’s license state
- **ABCompany** – Name, contact tag, and one of the following:
 - Primary address – Address line 1, city, state, ZIP code
 - Phone number – Any of the phone numbers, such as fax or work phone
 - Tax ID
- **ABCompanyVendor** – Name, contact tag, and tax ID
- **ABPersonVendor** – Last name, contact tag, and tax ID
- **ABPlace** – Name, contact tag, and primary address, consisting of address line 1, city, state, and ZIP code
- **ABUser** – Last name, contact tag, and employee number

Verify Minimum Criteria work queue

The Verify Minimum Criteria work queue enables you to iterate through a set of contacts that you have imported from staging tables to verify if they meet minimum creation criteria. The work queue uses the `ValidateABContactCreationPlugin` plugin interface.

The work queue iterates over all instances of `ABContact` and its subtypes in `ContactManager` for which verification has not yet passed, checking if `ABContact.MinimumCriteriaVerified` is `false`. For each unverified contact, the work queue calls the plugin implementation class registered with `ValidateABContactCreationPlugin.gwp`.

Note: In the base configuration, this class is `gw.plugin.contact.ValidateABContactCreationPluginImpl`.

If the plugin class indicates that the contact is valid, the work queue sets the contact's `MinimumCriteriaVerified` field to `true`.

The `MinimumCriteriaVerified` field is translated to XML when contacts are sent over web services. This field is exposed when the contact is obtained by calling `ABContactAPI.retrieveContact`, enabling a core application to determine if a contact has passed minimum validation criteria.

See also

- “Contact minimum creation requirements” on page 297

Configure the TaxID contact creation requirement

About this task

You can set whether or not TaxID is required for creating contacts. In the base configuration, this setting affects only `PersonVendor` and `CompanyVendor` subtypes. You set the variable `RequiresTaxID` in the registry file `ValidateABContactCreationPlugin.gwp`.

`ValidateABContactCreationPluginImpl` uses this value when it calls the following method that it inherits from `ValidateABContactCreationPluginBase`:

```
override function setParameters(p0: Map<Object, Object>) {
  _requiresTaxID = Boolean.valueOf(p0.get("RequiresTaxID") as String)
}
```

Procedure

1. If necessary, start Guidewire Studio™ for ContactManager.
2. In the **Project** window, navigate to **configuration**→**config**→**Plugins**→**registry** and double-click `ValidateABContactCreationPlugin.gwp`.
3. The default registered plugin implementation is in the **Gosu Class** field:
`gw.plugin.contact.ValidateABContactCreationPluginImpl`
4. Under the **Gosu Class** field, `RequiresTaxID` is defined in the **Parameters** table. The default value is `true`. If you set it to `false`, TaxID is not required for creating any contact subtype.

Code example for creating ABContact and ABCompany contacts

The following code sample from `ValidateABContactCreationPluginBase` shows code defining contact creation requirements for `ABContact` and `ABCompany`:

```
protected function abContactIsInvalid(contact : ABContact) : boolean {
  return contact.Tags == null or contact.Tags.isEmpty
}
protected function abCompanyIsInvalid(contact : ABCompany) : boolean {
  if (abContactIsInvalid(contact))
    return true
  if (RequiresTaxID) {
    return contact.Name == null
      or (isLackingCompleteAddress(contact.PrimaryAddress)
        and isLackingAnyPhoneNumber(contact)
        and contact.TaxID == null)
  } else {
    return contact.Name == null
      or (isLackingCompleteAddress(contact.PrimaryAddress)
        and isLackingAnyPhoneNumber(contact))
  }
}
```

Configuring validation error messages for contact creation

If an `ABContact` instance fails validation, `ValidateABContactCreationPluginImpl` throws `TooLooseContactCreateCriteriaException`, which supports a set of display keys based on locale and `ABContact` subtype.

You can use Guidewire Studio™ for ContactManager to access the set of display keys provided in the base configuration. For example, you can edit the U.S English display key properties file. Navigate in the **Project** window

to **configuration→config→Localizations→Resource Bundle 'display'** and double-click `display.properties` to open this file in the editor. In the editor, search for `TooLooseContactCreateCriteriaException`.

There is a set of `TooLooseContactCreateCriteriaException` display keys for `ABContact` and some of its subtypes. These display keys are accessible in code. For example:

```
Java.TooLooseContactCreateCriteriaException.ABCompany
Java.TooLooseContactCreateCriteriaException.ABCompany.ja_JP
Java.TooLooseContactCreateCriteriaException.ABCompanyVendor
Java.TooLooseContactCreateCriteriaException.ABCompanyVendor.ja_JP
```

For example, an `ABContact` that is an `ABCompanyVendor` with locale `en_US` fails validation. It throws an exception, using for its message the display key `Java.TooLooseContactCreateCriteriaException.ABCompanyVendor`. If the Japanese locale is specified, the exception uses the display key

`Java.TooLooseContactCreateCriteriaException.ABCompanyVendor.ja_JP`.

Similar behavior applies to `ABContact` entities that are of class `ABCompany` but not `ABCompanyVendor`. They use the `Java.TooLooseContactCreateCriteriaException.ABCompany` display keys.

See also

- For general information on working with display keys, see the *Configuration Guide*.

Testing clock plugin interface

IMPORTANT The `ITestingClock` plugin interface is supported only for testing on non-production development servers. Do not register an implementation of this plugin on production servers.

To test `ContactManager` behavior over a simulated long span of time, you can implement the `ITestingClock` plugin interface and programmatically change the system time to simulate the passing of time. For example, you can define a plugin implementation that returns the real time except in special cases in which you artificially increase the time to represent a time delay. The delay could be one week, one month, or one year.

Time must always increase, not go back in time. Going back in time is likely to cause unpredictable behavior in `ContactManager`.

See also

- For full information on using `ITestingClock`, see the *Integration Guide*.

Release note archive

This topic contains the release notes for previous versions of ContactManager. You can use these previous release notes to see which issues have been fixed and how features have changed from one release to the next.

IMPORTANT This topic contains upgrade information originally provided for earlier ContactManager releases. It might be superseded by later release notes or other upgrade documentation.

Guidewire ContactManager 9.0.5 Release Notes

©2001-2018 Guidewire Software, Inc.

For information about Guidewire trademarks, visit <http://guidewire.com/legal-notice>.

Guidewire Proprietary & Confidential — DO NOT DISTRIBUTE

These release notes contain the following topics:

- “Release information” on page 301
- “Installing this release” on page 302
- “Support” on page 302
- “Major issues and changes” on page 302
- “Improvements and resolved issues” on page 303
- “Known issues and limitations” on page 307

Release information

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped upgrading to one or more of the previous releases of ContactManager, be sure to read the *Release Notes* for those releases to learn about related changes and fixed issues.

Release number

This release of Guidewire ContactManager is 9.0.5.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 8.0.0 or later. The latest maintenance release is preferred.

Installing this release

Refer to the following:

- "Installing ContactManager" in the *Guidewire Contact Management Guide* for general installation information
- Prior ContactManager release notes for any releases that you have skipped

Upgrade information

Significant changes to upgrade procedure of ContactManager 9

IMPORTANT The procedure for upgrading to ContactManager 9 from ContactManager 8 or previous releases has changed significantly. In ContactManager 9, there are new tools available for performing the upgrade. Do not rely on your past knowledge of the upgrade process to begin the upgrade procedure. Before beginning your upgrade, review the topic "Configuration Upgrade Overview" in the *Upgrade Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire Customers

- <https://community.guidewire.com>

Guidewire Partners

- <https://partner.guidewire.com>

Major issues and changes

These release notes describe major issues and changes that can affect your installation.

For information on new features and major changes, see the topic "New and Changed in ContactManager 9.0.5" in the *Guidewire Contact Management Guide*.

Base PCF file changes

The link that follows requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

ContactManager release 9.0.4 to 9.0.5

To view a report of the changes to the base PCF files, *click here*.

Base rule changes

ContactManager release 9.0.4 to 9.0.5

There were no changes in ContactManager rules for this release.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community and search for knowledge article 7898, "How to find reports detailing differences between earlier and later releases of Guidewire core products".

Improvements and resolved issues

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ContactManager 905 improvements and resolved Issues

ID	Description
CTC-4093	Updated the Geocode plugin to support Bing Maps V8 with REST API.
CTC-4203	Fixed a bug involving change history for a phone number field. Change history for a phone number field now displays, even if the field's phonecountrycode property does not use the PhoneCountryCode typelist.
CTC-4257	A unique index on ABLinkable has been moved to extensions, making it possible to remove the index.

Platform improvements and resolved issues

ID	Description
PL-20685	Fixed an issue where a web service call resulted in a <code>java.lang.LinkageError</code> . The issue was resolved by removing the <code>JRE activation.jar</code> and <code>saa.jar</code> files from the Guidewire-packaged WAR files.
PL-30292	Fixed an issue that caused the Not Started column in the Consistency Check Runs report to always show 0.
PL-32345	Modified the on-screen labels and added an information message to improve the usability of the Consistency Check popup.
PL-35419	Fixed a message queue issue that occurred after losing and restoring a database connection. The resumed message queue status was shown as Started, but the queue did not process messages.
PL-35603	The unwanted test modules that were previously being packaged into the build are removed.
PL-35658	Guidewire has deprecated the <code>TableImportAPI</code> methods that perform work synchronously. Guidewire recommends that you use the version of the method that performs work asynchronously instead. In some cases, Guidewire added the asynchronous version of the method if it did not exist. For the asynchronous methods, use the <code>ProcessID</code> and poll using the Maintenance Tools API to determine if the work is complete.
PL-36179	Application logging during upgrade now includes upgrade information for troubleshooting purposes.
PL-36280	Fixed an issue where upgrade failed for some databases using a non-English language with following error: The conversion of a varchar data type to a datetime data type resulted in an out-of-range value
PL-36504	Fixed an issue that permitted the successful execution of staging table import at the DAEMONS server run level, which Guidewire does not support. Now, it is not possible to execute a staging table import in a server run level that is higher than maintenance (NODAEMONS).
PL-36547	The <code>BatchProcessBase#incrementOperationsFailedReasons</code> method is now null-safe.

ID	Description
PL-36577	Fixed an issue that caused a large number of Xml.Error-type messages when using a localized version of the xmlcodegen_en.properties file.
PL-36744	Fixed an issue where a SOAP MTOM (Message Transmission Optimization Mechanism) response that contained multiple sequential CR-LF pairs would throw an exception.
PL-36754	Corrected an issue that caused the Load History download to show incorrect or missing information.
PL-36924	In previous releases, a full upgrade of configuration changes only on a single, standalone ContactManager server would complete successfully. However, in that case, ContactManager did not update the Server Tools Upgrade and Versions screen with upgrade information. Guidewire has changed this behavior so that a full upgrade of a standalone server, even if only for configuration changes, now updates the upgrade information on the Upgrade and Versions screen.
PL-36961	Modified handling of LOB objects so as to make better use of Oracle temporary tablespace.
PL-37004	Made performance improvements to the database upgrade step that populates entity search denorm fields during the application upgrade.
PL-37029	Obfuscating a User entity by using PersonalDataDestructionAPI no longer causes the DBCC tool to report that the UserSetting object related to that User is missing.
PL-37034	Fixed an issue in which Oracle parallelism was not enabled for loader callbacks during the execution of the database integrityCheckAndLoad process. Also, updated Javadoc for DML methods for Loader callbacks to remind developers to use the methods in the class hierarchy that respect the parallel setting in the database-config <loader> and <callback> elements.
PL-37048	Removed the obsolete schemalocations.xml file, which was replaced with gwxmlmodule.xml in version 9.0.0.
PL-37057	Fixed an issue where running the command gwb genAllWsd1 failed with a ConcurrentModificationException.
PL-37072	Fixed an issue that would cause a server error when a startable plugin or external process called getRunLevel during server startup.
PL-37122	Addressed security issues by upgrading several third-party libraries.
PL-37135	Fixed an issue in JMS inbound integration with the ordered parameter set to false where polling an empty thread did not close the JMS session.
PL-37146	Any change to the useoraclestatspreferences attribute in database-config.xml (from false to true or from true to false) takes effect only after an upgrade, either a full upgrade or a rolling (configuration) upgrade. However, if you reset this attribute from true to false, ContactManager throws an exception during the next upgrade and prevents the upgrade from continuing due to locked table statistics in the Oracle database. Review the details of the exception provided in the server log to determine which table statistics need to be unlocked. See “Revert to DBStats Batch Processing for Database Statistics” in the <i>System Administration Guide</i> for information on how to unlock the table statistics.
PL-37153	Fixed an issue in which the DropForeignKeysToProcessHistoryAndDeleteProcessHistoryRecordsTrigger upgrade trigger deleted ProcessHistory data.
PL-37178	Guidewire has updated the ping utility, an unauthenticated web page that you can ping to access information about a ContactManager server. See the Guidewire documentation for the meaning of the fields that the ping utility returns.
PL-37186	Added a new public method to gw.api.profiler.Profiler: public static <V> Callable<V> createPotentiallyProfiledCallable(ProfilerTag entryPointTag, String entryPointDetail, Callable<V> block)
PL-37187	Fixed an issue related to Tomcat version 8.0.37 and higher for which it was possible to see a number of cache-related warning messages in the logs.

ID	Description
PL-37247	Fixed an issue that could cause an exception if bundle contents changed during iteration of the beans in the bundle. The <code>getBeansByRootType</code> method now returns a collection that is an immutable copy of the bean values in the bundle.
PL-37256	Previously, during database upgrade, table aliases had a numeric suffix that was incremented globally. As a result, depending on the starting point, two runs of the database upgrade could result in two different strings for the same query. This behavior has been updated so that table aliases are now consistent—running database upgrade on the same initial database does not result in two different strings for the same query.
PL-37263	You can now update the database statistics settings from file <code>database-config</code> with a rolling upgrade. This change can also include using <code>database-config</code> to switch to using Oracle statistics preferences.
PL-37266	Eliminated a security risk. Gosu servlets no longer accept a session ID if the HTTP connection is non-secure. For network topologies that secure the connection through other means, such as by not allowing external access to the server, the security check can be overridden by setting the system property <code>gw.servlet.ServletUtils.BypassIsSecure</code> to <code>true</code> .
PL-37284	Fixed an issue in JMS inbound integration that prevented recovery of the message queue if the connection to the application server failed.
PL-37292	Fixed an issue that caused the file inbound integration to stop processing files if the incoming folder contained more than 300 files.
PL-37300	The default configuration's <code>JavaxEmailMessageTransport</code> plugin processes an HTML or plain text email as intended. In the plugin's <code>addBody</code> method, the email is sent as HTML or plain text based only on the value of the <code>email</code> argument's <code>Html</code> property, which has a default value of <code>false</code> . In the default configuration, the content of the email is of unknown origin. Scanning the content to determine whether it is HTML introduces a security risk. Specifically, the default configuration cannot determine whether the content includes attack-injected HTML. Configuration code that creates the email content itself, and therefore knows the content is secure, can explicitly set the <code>Html</code> property to <code>true</code> to have the email sent as HTML.
PL-37306	The database schema verifier no longer checks if the table name exceeds the Guidewire maximum table name length. This check occurs in the entity generator/verifier.
PL-37314	Fixed an issue in which duplicate IDs within a PCF could cause a <code>StackOverflowError</code> error while executing the <code>genDataDictionary</code> command.
PL-37315	Commented out an unnecessary print statement in <code>GwXmlElementEnhancement.gsx</code> .
PL-37322	Fixed an issue in JMS inbound integration that prevented the message queue from restarting if the destination shut down and restarted.
PL-37324	Improved the performance of foreign key constraint creation during upgrade.
PL-37363	Fixed an issue that caused <code>ContactManager</code> to throw a <code>NullPointerException</code> exception if method <code>EmailContact.setContact(contact : Contact)</code> had a null <code>Contact</code> as a parameter.
PL-37365	Changed the default value for <code>ora-parallel-dml</code> in the upgrade element in file <code>database-config</code> from <code>enable</code> to <code>enable-all</code> . This change means, in Oracle, that <code>INSERT</code> , <code>UPDATE</code> , and <code>DELETE</code> statements in version triggers run in parallel by default.
PL-37369	Fixed an issue that generated an exception in the server log if running <code>DBStats</code> or <code>DBConsistencyCheck</code> batch processing with no arguments from the command prompt.
PL-37384	Fixed an issue where database upgrade steps were triggered with <code>versioncheckonly=true</code> . The <code>versioncheckonly</code> setting now works as expected during database upgrade.
PL-37394	Improved the reporting of upgrade-related activity in the application log.
PL-37406	Improved information in the server log during an upgrade. For example, the log now states explicitly when the upgrade completed: Full upgrade completed successfully: schema and configuration changes have been applied
PL-37429	Fixed an issue that caused a <code>Null Pointer Exception</code> during the execution of an Oracle AWR report.

ID	Description
PL-37447	Addressed security issues by upgrading the poi-ooxml and fop JAR libraries. A new and empty FOPPrint.config file was added to the config/resources directory.
PL-37448	Eliminated a security risk by updating the ant and ant-junit JAR files.
PL-37461	Implemented performance improvement around server startup.
PL-37471	Eliminated a potential denial-of-service security risk by updating the commons-fileupload library. The library is used when uploading files during the document management process.
PL-37475	Fixed an issue with a document template not being able to load a dynamic image.
PL-37486	Fixed an issue where a web service client making an external API call would not release its socket when timed out.
PL-37498	Fixed an issue in a rolling upgrade that caused ContactManager to not properly mark deferred tasks as completed. This issue prevented archiving from running.
PL-37530	Fixed a concurrency issue with the gw.xml.ws.AsyncResponse.get method.
PL-37561	Changed the message-locking behavior for distributed message transactions. When processing a distributed message transaction, the transaction's message object is not locked. This locking behavior can be configured by setting the new LockDuringDistributedMessageRequestHandling configuration parameter. The message-locking behavior for non-distributed message transactions remains unchanged. For non-distributed transactions, the message object is always locked. If the message object is not locked, any associated primary entity will not be locked either, even if locking of the entity is enabled. For details, refer to the main documentation.
PL-37592	Fixed an issue that caused the rolling upgrade process to not accept changes to configuration parameters listed as local, and, instead, forced a full upgrade.
PL-37600	Fixed an issue that caused duplicate database index definitions for an index defined on an entity delegate extension. This issue caused the index definition to fail.
PLWEB-6332	Fixed an issue where a BooleanRadioInput that toggled the visibility of other screen elements would cause excessive screen flickering.
PLWEB-6334	Fixed an issue in a time field where entering an invalid value would not be validated and the value would be discarded.
PLWEB-6337	Fixed an issue where clicking on a field when an off-screen field had focus would cause the screen to jump to the field with previous focus.
PLWEB-6338	Fixed an issue where selecting multiple rows in two separate list views and then clicking Remove would remove the rows from both list views. Instead, only the rows in the list view containing the clicked button are removed.
PLWEB-6629	Fixed an issue where you could not select the title text in a wizard widget.
PLWEB-6630	Pressing the AltGr key no longer behaves as if the Alt key is pressed.
PLWEB-6631	Fixed an issue where validation messages were not visible when they appeared in a Workspace panel that was hidden. If the Workspace is hidden when new messages appear, the Workspace now opens automatically.
PLWEB-6643	When running the application from the command line, the ability to dynamically reload PCF files is disabled by default. You can restore it by adding the -DgosuInit.supportDiscretePackages=true VM option to the command line call. This functionality is retained by default when running the application from Guidewire Studio by the presence of this option in the Studio server configuration.
PLWEB-6649	Fixed an issue where check box changes on newly expanded RowTree rows were ignored when submitting.
PLWEB-6651	When specifying a number as a keyboard shortcut, both the main keyboard number key as well as the corresponding key on the number pad now work as that shortcut.
PLWEB-6652	The required attribute of a Cell can no longer access the iterated value.
PLWEB-6653	Fixed an issue where a Link with an Arg was not shown.

ID	Description
PLWEB-6658	Fixed an issue where the wrap attribute on a Cell was not respected by FormalCell or LinkCell.
PLWEB-6660	Restored attribute reflectOnBottom to the PCF widget Toolbar.
PLWEB-6670	Added attributes actionHandler and updateHandler to PCF widget TemplatePanel.
PLWEB-6671	Added new PCF widget PanelIteratorRemoveLink. The PCF editor in Guidewire Studio does not yet fully support this new widget, so you might need to manually edit the XML of the PCF file to add PanelIteratorRemoveLink.
PLWEB-6685	Fixed an issue where collapsing an input group located at the bottom of a scrolled screen caused the view to shift.
PLWEB-6694	Fixed an issue where a list detail view would not show the correct pagination when selecting a new row.
PLWEB-6842	Fixed an issue with unnecessary locking of PCF files, which caused a performance issue in PCFServiceImpl.
PLWEB-6851	Fixed an issue with type system locking that impacted the performance of rendering PCF files.

Known issues and limitations

This section describes known issues with this release of Guidewire ContactManager.

Note: For maintenance releases, Guidewire often defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue

Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one ContactContact entity. ContactManager does not verify that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant ContactContact entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround

In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one ContactContact entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue

Using the RelatedABContactSearchCriteria entity and the CriteriaDef elements that are defined for it in search-config.xml is not supported in this version of ContactManager.

Workaround

Do not use them. Guidewire is aware of this issue.

GDPR - Not returning ManualInterventionRequired status when an exception thrown during purging (CTC-4150)

Issue

The `ABPersonalDataDestroyer` class does not return the correct status when an unknown exception is caught.

Workaround

Open the `ABPersonalDataDestroyer` class in Guidewire Studio and make the following changes:

In private function `destroyContact(contact: ABContact)`, change the return value of the following code:

```
} catch (e: Exception) {  
  ABPersonalDataLogUtil.logErrorNotDestroyed(contact, e)  
  return ContactDestructionStatus.TC_NOTDESTROYED
```

Change the return value to:

```
return ContactDestructionStatus.TC_MANUALINTERVENTIONREQUIRED
```

Studio/Platform known issues

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue

If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround

Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

Alignment Property Does Not Work for Inputs in a DetailView (PL-29429)

Issue

Inputs in a detail view do not align to the right or center. They always align to the left, regardless of the value of the `align` property.

Workaround

If you need to right align a series of items such as a set of monetary values, put them into a list view instead of a detail view. You can then set the alignment as needed.

JMS inbound integration implementations must be OSGi or Java plugins (PL-32054)

Issue

Writing JMS inbound integration handlers requires casting an object to the type `javax.jms.Message`. In Gosu, for some app servers this fails with the message: `Caused by:`

```
gw.lang.parser.exceptions.ErrantGosuClassException: GosuClass  
test.GosuJMSMessageReplyHandler has errors, and cannot be used at runtime.
```

Workaround

Implement JMS handler code for inbound integration in Java. Guidewire recommends that for inbound integration Java code, you implement your code as an OSGi plugin, not as a standard (non-OSGi) Java plugin.

Compilation errors after removing a delegate (PL-34619)

Issue

When you remove a delegate, some internal code is not generated properly, resulting in compilation errors.

Workaround

In Guidewire Studio, on the **Codegen** menu, click **Generate Metadata Classes**.

Several permissions are defined incorrectly (PL-34946)

Issue

The following permissions are defined incorrectly:

- perm.Holiday.edit
- perm.Holiday.delete
- perm.Note.view

Workaround

To any role that contains the `holidaymanage` permission, add the `buswkmanage` permission to that role. To any role that contains the `noteview` permission, add the `noteedit` permission to that role.

Alt+Shift+L to reload PCF files does not work when the server is run from the command line (PLWEB-5819)

Issue

If you run the server from the command line with the command `gwb runServer`, then pressing `Alt+Shift+L` in the application interface does not reload PCF files.

Workaround

Run the server from Guidewire Studio instead.

Server Tools downloads might time out (PLWEB-6184)

Issue

On the **Server Tools** pages, some downloads might take too long and generate a timeout error.

Workaround

Increase the value of the `WebUIAJAXTimeout` configuration parameter.

Batch process freezes if checkInitialConditions returns false (PL-36852)

Issue

If the `BatchProcessBase.checkInitialConditions` method returns `false`, the application freezes any ongoing work in that batch process. It then continues to retry the `checkInitialConditions` method for a period of 10 minutes until either the check returns `true` or the time limit expires. Although this is the expected behavior, it is possible for the recurring execution of the `checkInitialConditions` method to cause performance problems. This is also a change from the behavior of this method in releases prior to the 9.0.0 release.

Workaround

Add code similar to the following to the batch process implementation class. The example code may need some modification for a custom work queue writer.

```
private var _readyToRun = true

override function checkInitialConditions() : boolean {
    _readyToRun = ... // verify initial conditions here or in delegated method
    return true
}

override function doWork() {
    // verify initial conditions here and short-circuit method if conditions are not met
    if(!_readyToRun) {
        setOperationsFailedReasons( { "No work" } )
        return
    }
    // body of batch process follows
}
```

```
} ...
```

Apache ZooKeeper security issue CVE-2017-5637 (PL-37446)

Issue

Apache ZooKeeper has released a security update (CVE-2017-5637) for its server. Guidewire Solr Extension code uses only the client side of the ZooKeeper software, not the server side. As such, the underlying security issue that ZooKeeper has addressed does not affect Guidewire applications directly.

Workaround

If you wish to incorporate the ZooKeeper security update, Guidewire recommends that you update the server-side ZooKeeper software from version 3.4.6 to 3.4.10.

Guidewire ContactManager 9.0.4 release notes

These release notes contain the following topics:

- “Release information for ContactManager 9.0.4” on page 310
- “Installing ContactManager 904” on page 310
- “Support” on page 21
- “Major issues and changes for ContactManager 9.0.4” on page 311
- “Improvements and resolved issues for ContactManager 9.0.4” on page 311
- “Known issues and limitations for ContactManager 9.0.4” on page 314

Release information for ContactManager 9.0.4

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the Release Notes for those releases to learn about changes and fixed issues. See “Release note archive” on page 301.

This release of Guidewire ContactManager is 9.0.4.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 8.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 904

For general installation information, see “Installing ContactManager” on page 55.

For versions prior to 9.0.4 that you have skipped:

- For versions subsequent to 7.0.0, see “Release note archive” on page 301.
- For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include topics for ContactCenter release notes.

Upgrade information for ContactManager 9.0.4

Significant Changes to upgrade procedure of ContactManager 9

IMPORTANT The procedure for upgrading to ContactManager 9 from ContactManager 8 or previous releases has changed significantly. In ContactManager 9, there are new tools available for performing the upgrade. Do not rely on your past knowledge of the upgrade process to begin the upgrade procedure. Before beginning your upgrade, review the topic “Configuration Upgrade Overview” in the *Upgrade Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 9.0.4

These release notes describe major issues and changes that can affect your installation.

Base PCF file changes for ContactManager 9.0.4

The link in the original release notes requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

ContactManager release 9.0.3 to 9.0.4

To view a report of the changes to the base PCF files, refer to the original ContactManager 9.0.4 release notes.

Base rule changes for ContactManager 904

ContactManager release 9.0.3 to 9.0.4

There were no changes in ContactManager rules for this release.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 9.0.4

The topics that follow describe improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers, and the list is not intended to be comprehensive.

ContactManager 904 improvements and resolved issues

ID	Description
Activities, Activity Patterns, Workplan	
CTC-4093	Updated the Geocode plugin to support Bing Maps V8 with REST API.

Platform improvements and general issues for ContactManager 904

The following are the primary improvements and issues corrected in this release.

ID	Description
Database	
PL-36421	Fixed an issue in which log messages from database startup and upgrade contained a mixture of hard-coded English and the localized language.
Database Instrumentation	
PL-36363	Fixed an intermittent issue with Oracle AWR reports caused by the use of an incorrect datatype.
PL-36679	Correctly ordered the column headings in the Message Queue Depth report in the AWR download on the Server Tools page.
Database Support - General	
PL-34712	If using ContactManager to update database statistics on Oracle or SQL Server, the configuration now allows an option to specify force as a table action. ContactManager updates statistics if running incremental statistics regardless of the percentage threshold.
PL-34965	Removed the databasedegree and samplingpercentage attributes from the histogramstatistics element in database-config.xml. It has no meaning or effect at the column (histogram) level.
Database Support - Oracle	
PL-36624	Fixed an issue that caused an Oracle error when using the browser Find to search localized text in a long note.
Database Upgrade	
PL-36161	The database upgrade step of migrating a SQL Server database from using four-byte integers and the datetime datatype to eight-byte big integers and the datetime2 datatype has been rewritten to run much faster. Note: This conversion will not be available in 10.0.x releases and must be done while still on a 9.0.x release.
PL-36291	Fixed an issue where upgrading a SQL Server database from 7.0 to 9.0 caused a data truncation error.
Document Management	
PL-36623	Fixed an issue where Note links to a document were broken if the IDocumentMetadataSource plugin was enabled.
PL-36738	Fixed an issue with the rendering of documents linked in the body of a note. When retrieving such documents, the IDocumentContentSource plugin might ignore the TargetHiddenFrame property of the DocumentContentsInfo object. The TargetHiddenFrame property specifies whether the document web page is opened in a hidden browser frame.
Email	

ID	Description
PL-36644	Fixed a NullPointerException issue when processing an email in the MessageTransport plugin. The issue occurred whenever the recipient's email address was invalid.
Entities/Metadata	
PL-36423	You can now enable histograms on base typekey columns by using the new createhistogram attribute of the <typekey-override> element.
PL-36818	Fixed an issue that occurred after merging the product and data models using the Guidewire Upgrade tools to upgrade to version 9.0.x. An exception was thrown when the Studio Codegen → Generate Everything menu item was selected.
PL-36838	The description of the Contact entity in the Data Dictionary now displays correctly.
Gosu	
GOSU-31	Studio and the Gosu compiler now enforce the restriction that methods, including property accessor methods, cannot be both final and static.
GOSU-32	Fixed an issue where a BigDecimal to MonetaryAmount comparison caused an error.
GOSU-33	Fixed an issue in auto-casting to structural types.
GOSU-34	Fixed a Gosu parser issue that caused a compilation error.
GOSU-38	Fixed an issue that prevented the Studio debugger from evaluating some expressions.
GOSU-40	Fixed an issue in compiling Gosu code that used the compound assignment operator and dimensions.
GOSU-42	Fixed an issue that created incorrect byte-code for enhancement methods on Gosu interfaces.
PL-36712	Fixed an issue that prevented a protected inner enumeration being visible to a subclass of the enclosing class.
PL-36799	Fixed an issue that prevented a protected inner class being visible to a subclass of the enclosing class.
Infrastructure	
PL-36755	Fixed an issue where the ContactManager ready message appeared before the user could log in.
Inbound Integration	
PL-36630	Fixed an issue where file-based Inbound Integration did not process files that existed in the incoming directory at server startup.
Integration	
PL-36643	Fixed an issue where the suite-config.xml file's gw.cc.env parameter would be ignored.
PL-36699	Fixed an issue with the gwb exportwsdl command where nested XSD files were not included in the generated JAR file.
Messaging	
PL-36546	Fixed an issue where the message queue status is shown as Started even though the message queue did not start.
PL-36645	Fixed an issue where paging was not enabled to scroll through the entries on the Administration → Monitoring → Message Queues → Destination screen.

ID	Description
PL-36696	To improve troubleshooting operations, all failures to suspend or resume a message destination are logged.
Metadata Code Generator	
PL-36652	Fixed an error that occurred when a typelist had the same name as an entity sub-type.
PL-36748	Fixed an issue where attempting to remove an index using the <remove-index> element would result in an error that the index "is not perf-only".
PCF - Widget - Address AutoFill	
PLWEB-6335	Fixed an issue where an invalid postal code would sometimes not be highlighted.
PCF - Widget - Menu	
PLWEB-6318	Fixed an issue with ListView header menus not opening when using a Microsoft Surface device.
PCF - Widget - Modal Cell	
PLWEB-6336	Fixed an issue where text would not wrap in a modal cell in a list view.
Persistence	
PL-36585	Fixed an issue where incorrect Java classes would be generated when an entity implemented a structure.
PL-36813	Fixed an issue where running gwb genDataMapping would not populate the Description column in the resulting DataMap.csv file.
Rolling Upgrade	
PL-36333	<p>Added an autoupgrade attribute to the <database> element in file database-config.xml. This attribute takes two values:</p> <ul style="list-style-type: none"> • full: Full takes precedence and initiates a full upgrade assuming all other necessary conditions are met. • manual: Requires that you set either the database upgrade type (in Server Tools Upgrade and Versions screen) or the date system property.
Search	
PL-34987	Updated HttpClient library to 4.5.3.
Web - UI/Runtime	
PLWEB-6317	Fixed an issue that prevented an ExitPoint from working within a worksheet.
PLWEB-6360	Fixed an issue where the error message Please check server log was shown, but there was no information added to the log.
Web Services - WSI (New)	
PL-36659	Fixed an issue where a web service call might be directed to an incorrect web service provider. The issue could occur if the definition of a <webservice-collection> included an <override-url> element that did not specify an <env> element. In such cases, the incorrect <url> element might be selected.

Known issues and limitations for ContactManager 9.0.4

There are known issues with this release of Guidewire ContactManager.

Note: Guidewire sometimes defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 9.0.4 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one ContactContact entity. ContactManager does not verify that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant ContactContact entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one ContactContact entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the RelatedABContactSearchCriteria entity and the CriteriaDef elements that are defined for it in search-config.xml is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

GDPR - Not returning ManualInterventionRequired status when an exception thrown during purging (CTC-4150)

Issue – The ABPersonalDataDestroyer class does not return the correct status when an unknown exception is caught.

Workaround – Open the ABPersonalDataDestroyer class in Guidewire Studio and make the following changes: In private function destroyContact(contact: ABContact), change the return value of the following code:

```
} catch (e: Exception) {  
  ABPersonalDataLogUtil.logErrorNotDestroyed(contact, e)  
  return ContactDestructionStatus.TC_NOTDESTROYED
```

Change the return value to:

```
return ContactDestructionStatus.TC_MANUALINTERVENTIONREQUIRED
```

Studio/Platform known issues for ContactManager 9.0.4

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the http:// protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the file:// protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the pdf subdirectory of the doc directory.

Alignment property does not work for inputs in a DetailView (PL-29429)

Issue – Inputs in a detail view do not align to the right or center. They always align to the left, regardless of the value of the `align` property.

Workaround – If you need to right align a series of items such as a set of monetary values, put them into a list view instead of a detail view. You can then set the alignment as needed.

JMS inbound integration implementations must be OSGi or Java plugins (PL-32054)

Issue – Writing JMS inbound integration handlers requires casting an object to the type `javax.jms.Message`. In Gosu, for some app servers this fails with the message: “Caused by: gw.lang.parser.exceptions.ErrantGosuClassException: GosuClass test.GosuJMSMessageReplyHandler has errors, and cannot be used at runtime.”

Workaround – Implement JMS handler code for inbound integration in Java. Guidewire recommends that for inbound integration Java code, implement your code as an OSGi plugin, not a standard (non-OSGi) Java plugin.

Compilation errors after removing a delegate (PL-34619)

Issue – When you remove a delegate, some internal code is not generated properly, resulting in compilation errors.

Workaround – In Guidewire Studio, on the **Codegen** menu, click **Generate Metadata Classes**.

Several permissions are defined incorrectly (PL-34946)

Issue – The following permissions are defined incorrectly:

- `perm.Holiday.edit`
- `perm.Holiday.delete`
- `perm.Note.view`

Workaround – To any role that contains the `holidaymanage` permission, add the `buswkmanage` permission to that role. To any role that contains the `noteview` permission, add the `noteedit` permission to that role.

Alt+Shift+L to reload PCF files does not work when the server is run from the command line (PLWEB-5819)

Issue – If you run the server from the command line with the command `gwb runServer`, then pressing Alt+Shift+L in the application interface does not reload PCF files.

Workaround – Run the server from Guidewire Studio instead.

Server Tools downloads may time out (PLWEB-6184)

Issue – On the **Server Tools** pages, some downloads may take too long and generate a timeout error.

Workaround – Increase the value of the `WebUIAJAXTimeout` configuration parameter.

During rule import, selecting the deployed rule results in incorrect rule version and status (BIZ-912)

Issue – Under certain circumstances, the business rule import process sets the rule version incorrectly. This occurs in the following circumstances:

- The existing and importing rules show as a version conflict.
- The existing or importing rule version is in the deployed state.
- The other rule version has a status other than deployed (draft, staged, approved).
- The deployed version of the rule is selected.

In all of these cases, the rule version is set to Approved with the deployed rule version number. For example, the deployed version is 1 and the version after import is 1+ Approved.

Workaround – Guidewire is aware of this issue.

DBCC scan reports obfuscated users (BC-16712)

Issue – In the base configuration, personal data destruction of a user results in the obfuscation of the User entity and the removal of the linked UserSettings entity from User. A subsequent database consistency check reports a

constraint violation for such users because it is unable to find the `UserSettings` entity. The issue is reported in BC-16712, but it applies to all Guidewire InsuranceSuite applications.

Workaround – If you do not enable personal data destruction, this constraint violation will not occur. If you do enable personal data destruction, you can ignore these constraint violations reported by the database consistency checking tool DBCC.

Guidewire ContactManager 9.0.3 release notes

These release notes contain the following topics:

- “Release information for ContactManager 9.0.3” on page 317
- “Installing ContactManager 9.0.3” on page 317
- “Support” on page 21
- “Major issues and changes for ContactManager 9.0.3” on page 318
- “Improvements and resolved issues for ContactManager 9.0.3” on page 318
- “Known issues and limitations for ContactManager 9.0.3” on page 320

Release information for ContactManager 9.0.3

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the Release Notes for those releases to learn about changes and fixed issues. See “Release note archive” on page 301.

This release of Guidewire ContactManager is 9.0.3.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 8.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 9.0.3

For general installation information, see “Installing ContactManager” on page 55.

For versions prior to 9.0.3 that you have skipped:

- For versions subsequent to 7.0.0, see “Release note archive” on page 301.
- For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include topics for ContactCenter release notes.

Upgrade information for ContactManager 9.0.3

Significant Changes to Upgrade Procedure of ContactManager 9

IMPORTANT The procedure for upgrading to ContactManager 9 from ContactManager 8 or previous releases has changed significantly. In ContactManager 9, there are new tools available for performing the upgrade. Do not rely on your past knowledge of the upgrade process to begin the upgrade procedure. Before beginning your upgrade, review the topic “Configuration Upgrade Overview” in the *Upgrade Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 9.0.3

These release notes describe major issues and changes that can affect your installation.

New Studio options for managing application upgrades (IDE-3811, IDE-3960)

In the Guidewire Studio™ **Settings** dialog, on the **Gosu Compiler** screen, there are new options for helping you manage application upgrades more efficiently. You can disable local Gosu compilation, exclude compilation of tests, and treat PCF code generation errors as warnings. See the *Configuration Guide*.

Improve build performance by preprocessing XML schema files (PL-36353)

This release includes significant improvements to build times. In addition to the many built-in improvements, build time can be further reduced by preprocessing XML schema files and certain other resources. The preprocessed resources are stored in a JAR file, which is used in subsequent builds without the need to be reprocessed.

For details on integrating this feature in the build process, visit the Guidewire Community and search for knowledge article 7356, “Generating an XML Schema Jar”.

Base PCF file changes for ContactManager 9.0.3

The following link requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

[ContactManager release 9.0.2 to 9.0.3](#)

To view a report of the changes to the base PCF files, refer to the original ContactManager 9.0.3 release notes.

Base rule changes for ContactManager 9.0.3

[ContactManager release 9.0.2 to 9.0.3](#)

There were no changes in ContactManager rules for this release.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 9.0.3

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

Platform improvements and general issues for ContactManager 9.0.3

The following are the primary improvements and issues corrected in this release:

ID	Description
AppServer Support - JBoss	
PL-36302	Added support for JBoss Enterprise Application Platform version 7.0.2.
Build Infrastructure	
PL-36368	After adding a new GX model, WSDL, or XSD file, the subsequent compilation process has been optimized. The performance gain is achieved by significantly reducing the number of files processed during the compilation.
Database Support - Oracle	
PL-36274	Added the following option to the list of options to use in generating a set of performance reports from the Server Tools Oracle AWR Information screen: Include native Oracle report . This option ensures that ContactManager generates an Oracle Standard AWR report automatically as you generate a ContactManager AWR report, if the database user has been granted the required privileges. See “Oracle AWR” in the <i>System Administration Guide</i> for details.
Document Management	
PL-36424	When a document is retrieved from the document management system, the resulting Document object must include the retrieved metadata. Implementations of the Document Metadata Source plugin method <code>retrieveDocument</code> must explicitly restore the document's original internal values by calling the <code>DocumentsUtilBase.initOriginalValues</code> method. The original internal values are evaluated by subsequent permission checks. Permission handlers evaluate properties to determine whether the user has the appropriate permission to access the document. For example, an Edit handler checks the Document object's Status property and prohibits editing if the status is final.
GX Tools	
PL-36434	The <code>gwb verifyResources</code> command now verifies GX model files, workflow XML files, and WS-I webservice annotations, as well as PCF files, Gosu types, and XML schemas.
Gradle Build Infrastructure	
PL-36401	You can now adjust memory settings for various command-line processes in the <code>gradle.properties</code> file. See “Tune Command Line Tool Memory Settings” in the <i>Installation Guide</i> .
PL-36511	Fixed an issue where generating a WAR file would produce an invalid file if any of the files in the configuration had names containing non-ASCII characters.
PL-36550	The <code>gwb studio</code> command automatically runs code generation first if it has not yet been run.
Inbound Integration	
PL-34046	Fixed an issue that resulted in an error when changing the server run level from maintenance to multiuser.
Integration	
PL-34510	In the <code>ImportResults.pcf</code> file, the value of the <code>canVisit</code> property has been changed from <code>perm.User.exportadmindata</code> to the correct <code>perm.User.importadmindata</code> .
Other	
PLWEB-6287	Replaced several hard-coded strings in JavaScript files with display keys.
PCF - Actions - Auto Complete	
PLWEB-6286	Fixed an issue where you could not type into a TaxID or SSN text box after navigating from a field that used auto-completion.
PCF - Layout - List View - Iterator	
PLWEB-6261	Fixed an issue where exporting CSV data from a LinkIterator widget would produce empty values.
Rolling Upgrade	
PL-36406	Fixed an issue that caused rolling upgrade checksum verification to take longer than necessary on server start-up.

ID	Description
Web - PCF Compiler	
PLWEB-6263	Fixed an issue where the PCF compiler would not show an error when column level attributes were assigned row level values.
Web - PCF Compiler - Studio	
PLWEB-6269	Fixed an issue where PCF files would not reload when there was a compilation error in any Gosu code that was loaded as part of the page.

Known issues and limitations for ContactManager 9.0.3

This topic describes known issues with this release of Guidewire ContactManager.

Note: Guidewire sometimes defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 9.0.3 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not verify that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one `ContactContact` entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the `RelatedABContactSearchCriteria` entity and the `CriteriaDef` elements that are defined for it in `search-config.xml` is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

Studio/Platform issues for ContactManager 9.0.3

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

Alignment property does not work for inputs in a DetailView (PL-29429)

Issue – Inputs in a detail view do not align to the right or center. They always align to the left, regardless of the value of the `align` property.

Workaround – If you need to right align a series of items such as a set of monetary values, put them into a list view instead of a detail view. You can then set the alignment as needed.

JMS inbound integration implementations must be OSGi or Java plugins (PL-32054)

Issue – Writing JMS inbound integration handlers requires casting an object to the type `javax.jms.Message`. In Gosu, for some app servers this fails with the message: “Caused by: gw.lang.parser.exceptions.ErrantGosuClassException: GosuClass test.GosuJMSMessageReplyHandler has errors, and cannot be used at runtime.”

Workaround – Implement JMS handler code for inbound integration in Java. Guidewire recommends that for inbound integration Java code, implement your code as an OSGi plugin, not a standard (non-OSGi) Java plugin.

Compilation errors after removing a delegate (PL-34619)

Issue – When you remove a delegate, some internal code is not generated properly, resulting in compilation errors.

Workaround – In Guidewire Studio, on the **Codegen** menu, click **Generate Metadata Classes**.

Several permissions are defined incorrectly (PL-34946)

Issue – The following permissions are defined incorrectly:

- `perm.Holiday.edit`
- `perm.Holiday.delete`
- `perm.Note.view`

Workaround – To any role that contains the `holidaymanage` permission, add the `buswkmanage` permission to that role. To any role that contains the `noteview` permission, add the `noteedit` permission to that role.

Alt+Shift+L to reload PCF files does not work when the server is run from the command line (PLWEB-5819)

Issue – If you run the server from the command line with the command `gwb runServer`, then pressing Alt+Shift+L in the application interface does not reload PCF files.

Workaround – Run the server from Guidewire Studio instead.

Server Tools downloads may time out (PLWEB-6184)

Issue – On the **Server Tools** pages, some downloads may take too long and generate a timeout error.

Workaround – Increase the value of the `WebUIAJAXTimeout` configuration parameter.

Guidewire ContactManager 9.0.2 release notes

These release notes contain the following topics:

- “Release Information for ContactManager 9.0.2” on page 321
- “Installing ContactManager 9.0.2” on page 322
- “Support” on page 21
- “Major issues and changes for ContactManager 9.0.2” on page 322
- “Improvements and resolved issues for ContactManager 9.0.2” on page 323
- “Known issues and limitations for ContactManager 9.0.2” on page 325

Release Information for ContactManager 9.0.2

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the Release Notes for those releases to learn about changes and fixed issues. See “Release note archive” on page 301.

This release of Guidewire ContactManager is 9.0.2.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 8.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 9.0.2

For general installation information, see “Installing ContactManager” on page 55.

For versions prior to 9.0.2 that you have skipped:

- For versions subsequent to 7.0.0, see “Release note archive” on page 301.
- For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include topics for ContactCenter release notes.

Upgrade Information for ContactManager 9.0.2

Significant Changes to Upgrade Procedure

IMPORTANT The procedure for upgrading to ContactManager 9 from ContactManager 8 or previous releases has changed significantly. In ContactManager 9, there are new tools available for performing the upgrade. Do not rely on your past knowledge of the upgrade process to begin the upgrade procedure. Before beginning your upgrade, review the topic “Configuration Upgrade Overview” in the *Upgrade Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 9.0.2

This topic contains major issues and changes that can affect your installation.

New and changed features for ContactManager 9.0.2

There were no new or changed features for ContactManager 9.0.2. See “New and changed features in ContactManager” on page 23.

Free text search is supported on WebLogic

Free-text search is now supported on the WebLogic application server.

Base PCF file changes for ContactManager 9.0.2

The following link requires that the ReleaseNotes_files directory be on your local disk in the same directory as this release note file.

[ContactManager release 9.0.1 to 9.0.2](#)

To view a report of the changes to the base PCF files, refer to the original ContactManager 9.0.2 release notes.

Base rule changes for ContactManager 9.0.2

The following link requires that the ReleaseNotes_files directory be on your local disk in the same directory as this release note file.

[ContactManager release 9.0.1 to 9.0.2](#)

To view a report of the changes to the base rules files, refer to the original ContactManager 9.0.2 release notes.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 9.0.2

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ContactManager 9.0.2 improvements and resolved issues

ID	Description
Contact Change Management	
CTC-3898	The contact history records have been changed to show the user name rather than the display name for a given contact change. It is not possible to always show the display name, so this change provides more consistency.
Localization	
CTC-3903	The labels on the Pending Changes screen for update are now defined in display keys and can be localized by importing a language pack.
CTC-3943	The Description and Valid Until field labels on the Tracked Changes section of the Contact History page can now be localized.
CTC-3969	The Merge Contacts screen now displays region-specific fields for contacts.
Services	
CTC-3499	Under some circumstances, such as deleting an existing column from the vendor services CSV file and then importing it, an inaccurate error message would be displayed saying that the import failed. The vendor services import process actually would run and finish, skipping invalid service columns if there were any. Importing the CVS file now displays the correct result for this scenario, “Import successful”.

Platform improvements and resolved issues for ContactManager 9.0.2

The following are the primary improvements and issues corrected in this release:

ID	Description
Archiving	
PL-36139	Microsoft SQL Server 2014 introduced a new optimizer cardinality estimator. The use of this optimizer creates sub-optimal plans during archiving, which can eventually lead to deadlocks on SQL Server. Guidewire disables this optimizer by adding a database hint at the end of the archiving UPDATE statement. For a SQL Server database to use this hint, you must grant the sysadmin role to the database schema owner.
Build Infrastructure	
PL-34507	You can now export all WSI web service WSDL and related resources to a single JAR file. The new command <code>gwb exportwsdl</code> generates the JAR file to the <code>modules\configuration\build</code> directory.
PL-36217	Updated the <code>gwb zipChangedConfig</code> build command so that generated files are not included in the changed configuration Zip file.
Database Support - General	
PL-35933	Guidewire has removed the unused <code>numfreqvals</code> attribute on the <code><histogramstatistics></code> element in file <code>database-config.xml</code> .
Database Upgrade	
PL-36178	During upgrade of an Oracle database, if the option <code>defer-create-nonessential-indexes="true"</code> was used with the option <code>degree-parallel-ddl</code> set to some value greater than 1, an Oracle exception would occur. This problem has been fixed.
Entities/Metadata	
PL-33429	The method <code>setFieldValue</code> is deprecated and reserved for Guidewire internal use only.
Management Plugin	
PL-36018	Fixed an issue that blocked publishing several server MBeans if the <code>JMXManagementPlugin</code> was enabled.
Other	
PLWEB-6203	Fixed an issue where putting the server into maintenance mode would not show a message in the UI stating that the server is undergoing maintenance.
PCF - Layout - List View	
PLWEB-6231	Fixed an issue where the cursor focus on a page would skip cells in row iterator widgets.
PCF - Layout - List View - Iterator	
PLWEB-6189	Fixed an issue where the configuration parameter <code>ListViewPageSizeDefault</code> did not have an effect.
Web - Performance	
PL-35768	Fixed an integer overflow error in <code>BCPReplicate.java</code> .
Web - PCF Compiler - Studio	
PLWEB-6269	Fixed an issue where PCFs would not reload when there was a compilation error in any Gosu code that was loaded as part of the page.
Web - UI/Runtime	
PLWEB-6146	Fixed an issue where the <code>SelectOnEnter</code> property on a card would throw an exception.
PLWEB-6252	Fixed an issue where user interface shortcut keys would not work.
Web Services - WSI (New)	
PL-36114	The XML schema files <code>wsdl.xsd</code> and <code>soap11.xsd</code> were updated to correctly conform to the WS-I Basic Profile 1.1 specification.
PL-36127	If a web service includes the <code>@WsicheckDuplicateExternalTransaction</code> annotation then its <code>transaction_id</code> header will be exposed.

ID	Description
PL-36251	Fixed an issue in WSDL generation for services with multiple @Throws annotations for the same exception but different reasons. This issue caused an exception in some cases for WSDL consumers. The WSDL now merges related faults into one fault with all explanations concatenated in one comment. The fix prevents an exception that some WSDL consumers experienced.
Work Queues	
PL-36226	Fixed an issue in which class BulkInsertWorkitemBase did not use system time in creating work items.
Workflow	
PL-6352	Resource validation now flags additional workflow definition errors.
XMLElement (and XSD types)	
PL-36071	In Studio, you no longer need to explicitly run Codegen → Generate Xml Classes . Studio now includes XML classes when it generates internal code automatically.
PL-36146	An XSD file can define an <enumeration> element with a value attribute that includes Polish characters.

Known issues and limitations for ContactManager 9.0.2

This topic describes known issues with this release of Guidewire ContactManager.

Note: Guidewire sometimes defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 9.0.2 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one ContactContact entity. ContactManager does not verify that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant ContactContact entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one ContactContact entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the RelatedABContactSearchCriteria entity and the CriteriaDef elements that are defined for it in search-config.xml is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

SSN / TaxID field not editable if autocomplete triggered (CTC-2989)

Issue – The **SSN/Tax ID** field of a contact is not immediately editable after an address autocomplete populates an address field.

Workaround – After autocomplete fills the address field, the **SSN/Tax ID** field has a drop-down menu button to its right. Click the button to open the menu, and choose either **Delete** or **Enter New Value**.

Studio/Platform known issues for ContactManager 9.0.2

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

Alignment Property Does Not Work for Inputs in a DetailView (PL-29429)

Issue – Inputs in a detail view do not align to the right or center. They always align to the left, regardless of the value of the `align` property.

Workaround – If you need to right align a series of items such as a set of monetary values, put them into a list view instead of a detail view. You can then set the alignment as needed.

JMS inbound integration implementations must be OSGi or Java plugins (PL-32054)

Issue – Writing JMS inbound integration handlers requires casting an object to the type `javax.jms.Message`. In Gosu, for some app servers this fails with the message: “Caused by: `gw.lang.parser.exceptions.ErrantGosuClassException: GosuClass test.GosuJMSMessageReplyHandler has errors, and cannot be used at runtime.`”

Workaround – Implement JMS handler code for inbound integration in Java. Guidewire recommends that for inbound integration Java code, implement your code as an OSGi plugin, not a standard (non-OSGi) Java plugin.

Compilation errors after removing a delegate (PL-34619)

Issue – When you remove a delegate, some internal code is not generated properly, resulting in compilation errors.

Workaround – In Guidewire Studio, on the **Codegen** menu, click **Generate Metadata Classes**.

Several permissions are defined incorrectly (PL-34946)

Issue – The following permissions are defined incorrectly:

- `perm.Holiday.edit`
- `perm.Holiday.delete`
- `perm.Note.view`

Workaround – To any role that contains the `holidaymanage` permission, add the `buswkmanage` permission to that role. To any role that contains the `noteview` permission, add the `noteedit` permission to that role.

Alt+Shift+L to reload PCF files does not work when the server is run from the command line (PLWEB-5819)

Issue – If you run the server from the command line with the command `gwb runServer`, then pressing `Alt+Shift+L` in the application interface does not reload PCF files.

Workaround – Run the server from Guidewire Studio instead.

Server Tools downloads may time out (PLWEB-6184)

Issue – On the **Server Tools** pages, some downloads may take too long and generate a timeout error.

Workaround – Increase the value of the `WebUIAJAXTimeout` configuration parameter.

Compiling project from Studio causes rolling upgrade to fail (PL-36030)

Issue – After compiling your project from Studio, rolling upgrade reports a configuration mismatch, and that a full upgrade is required.

Workaround – Edit the file `ContactManager/modules/script/gw-build.gradle`. In that file, locate the following code block:

```
webapp {
  deployTemplates = project.rootProject.file('modules/script/deploy-templates')
  warIncludes['modules/configuration/plugins'] = file('plugins')
}
```

Replace the above code block with the following:

```
webapp {
  deployTemplates = project.rootProject.file('modules/script/deploy-templates')
  file('plugins')?.listFiles().each { File outerFile ->
    outerFile?.listFiles().each { File innerFile ->
      if(innerFile.name != 'idea-gclasses') {
        warIncludes['modules/configuration/plugins/' + outerFile.name + '/' + innerFile.name] =
innerFile
      }
    }
  }
}
```

“Could not index class” errors appear in logs with JBoss 7 (PL-36231)

Issue – When you deploy the application with JBoss 7, the log includes “Could not index class” errors and associated stack traces.

Workaround – You can safely ignore these messages.

Guidewire ContactManager 9.0.1 release notes

These release notes contain the following topics:

- “Release Information for ContactManager 9.0.1” on page 327
- “Installing ContactManager release 9.0.1” on page 327
- “Support” on page 21
- “Major issues and changes for ContactManager 9.0.1” on page 328
- “Improvements and resolved issues for ContactManager 9.0.1” on page 329
- “Known issues and limitations for ContactManager 9.0.1” on page 332

Release Information for ContactManager 9.0.1

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the Release Notes for those releases to learn about changes and fixed issues. See “Release note archive” on page 301.

This release of Guidewire ContactManager is 9.0.1.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 8.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager release 9.0.1

For general installation information, see “Installing ContactManager” on page 55.

For versions prior to 9.0.1 that you have skipped:

- For versions subsequent to 7.0.0, see “Release note archive” on page 301.
- For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include topics for ContactCenter release notes.

Supported Java versions for ContactManager 9.0.1

The following are the supported Java versions for this release:

- Java JDK 1.8.0_92
- DCEVM version Java 8 update 92, build 1 (optional)

Version support for future releases will be published on the Guidewire Community.

Install the latest Guidewire Studio update for ContactManager 9.0.1

About this task

You can now download and apply updates to Guidewire Studio without needing to upgrade the entire ContactManager application. Studio automatically checks for updates and alerts you when they are available. As part of your ContactManager installation or upgrade, you should check for the latest update to Studio.

Procedure

1. Install Guidewire Studio following the standard instructions in the *Installation Guide*.
2. Run Guidewire Studio.
3. In the **Help** menu, click **Check for Studio Update**.
4. If an update is available, download and apply it.

Upgrade Information for ContactManager 9.0.1

Significant Changes to Upgrade Procedure for ContactManager 9.0.1

IMPORTANT The procedure for upgrading to ContactManager 9 from ContactManager 8 or previous releases has changed significantly. In ContactManager 9, there are new tools available for performing the upgrade. Do not rely on your past knowledge of the upgrade process to begin the upgrade procedure. Before beginning your upgrade, review the topic “Configuration Upgrade Overview” in the *Upgrade Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 9.0.1

This topic contains major issues and changes that can affect your installation.

New and changed features for ContactManager 9.0.1

For information on new features and other changes, see the topic “New and Changed Features in ContactManager 9.0.1” in the *Guidewire Contact Management Guide*.

Base PCF file changes for ContactManager 9.0.1

The following link requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

[ContactManager release 9.0.0 to 9.0.1](#)

To view a report of the changes to the base PCF files, refer to the original ContactManager 9.0.1 release notes.

Base rule changes for ContactManager 9.0.1

The following link requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

[ContactManager release 9.0.0 to 9.0.1](#)

To view a report of the changes to the base rules files, refer to the original ContactManager 9.0.1 release notes.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 9.0.1

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ContactManager 9.0.1 improvements and resolved issues

ID	Description
Document Management	
CTC-3871	Asynchronous document content storage is correctly enabled in this release. In version 9.0.0, if errors occurred in initial synchronous storage, documents were discarded rather than sent asynchronously. The asynchronous document content store transport (<code>documentStoreTransport.gwp</code>) is now enabled in the Plugins registry. The asynchronous document store message destination (ID 324) is now enabled in the Messaging registry. The configuration now conforms to Sync-first configuration as described in the <i>Integration Guide</i> in “Asynchronous Document Content Storage”.
Merge Contacts	
CTC-3949	Fixed an issue where some local copies of contacts in ClaimCenter were not getting updated with the latest contact data after contacts were merged in ContactManager.
Miscellaneous	
CTC-3047	There was an issue with the highlighting of rows that were defined in the PCF widget RowTree. This issue has been fixed. A Row widget that is defined in a RowTree and has its highlighted property set to true now displays as highlighted.
CTC-3048	There was an issue with the target property of an Exit Point widget defined in a PCF file. The issue prevented the Exit Point from functioning properly in the web browser. This issue has been fixed.
CTC-3049	Resolved an issue with auto-fill of the time portion of date-time fields. These fields now properly fill the time automatically for any valid date entry, including dates that have a leading zero for the day or month or both.

Platform improvements and resolved issues for ContactManager 9.0.1

The following are the primary improvements and issues corrected in this release:

ID	Description
Business Rules	
BIZ-822	Fixed an issue where editing the display values for the existing typekeys in Parentheses.tti and RuleBooleanOperator.tti would change the translation of the Rule Condition into Gosu.
Database Instrumentation	
PL-35669	The Server Tools Database Statistics download report now contains the full SQL command used to generate the table statistics. The report displays the command in the DBMS_STATS.GATHER_TABLE_STATS column for each listed table.
PL-35708	Fixed an issue that reported the elapsed time for consistency checks in the application log as 0.000 seconds.
Database Support - Oracle	
PL-34858	Removed the <indexstatistics> element from file database-config.xml.
PL-34960	Removed the Update Statistics Statements for Indexes table report from the Server Tools Database Statistics download report.
PL-34962	Fixed an issue that caused a Null Pointer Error during download of the (Server Tools) Database Catalog Statistics Information report. Attribute numbuckets is now a required attribute on the <histogramstatistics> element in database-config.xml.
PL-35491	The Oracle database is now supported. For version information, visit the Guidewire Community and search for knowledge article 1005, "Supported Software Components".
Database Upgrade	
PL-34087	Previously, during an Oracle database upgrade, if a table with locked statistics caused upgrade to fail, the error was not found until the end of the database upgrade process. Database upgrade now checks for this problem at the beginning of the upgrade process, allowing a fix to be applied immediately. Note: See the <i>Upgrade Guide</i> for information on disabling statistics update for tables with locked statistics.
PL-34740	Database upgrade now checks for multiple foreign key constraints on a column when verifying that the physical database and the data model match. Previously, the schema verifier assumed that there was at most one foreign key constraint on a column. Now, if the schema verifier finds multiple foreign key constraints on a column, the unexpected foreign keys are removed and a warning about the mismatch is logged.
Document Management	
PL-34624	Fixed an issue with printing notes that have attached documents.
Entities/Metadata	
PL-35825	Fixed an error that would occur when existing subtypes had their typecode retired.
PL-36060	Fixed an issue where generating the Data Dictionary would fail if there were certain Gosu errors in PCF files.
GX Tools	
PL-35531	Fixed an issue with GX Model serialization of typekey types when the Incremental option is set to true.
Logging	
PLWEB-4353	When an exception occurs while rendering the UI, the error log now records the source location of the exception.
Messaging	

ID	Description
PL-34569	There is now support for optionally distributing payload transformation across many servers in the cluster. The new distributed payload transformation work runs as a process across potentially multiple servers in the cluster, potentially long before the actual send time of the message. You can enable and configure this feature separately for each messaging destination. As part of this new feature, there is a new Message property called Bound, which is a boolean flag that specifies whether the message request processing is complete. You can optionally set it after message creation if you know that the message does not need processing. After the server performs the before send processing by calling the appropriate beforeSend method, the server sets Message.Bound to true. To correctly handle message resending, your beforeSend method must check Message.Bound and behave accordingly because the message may have the Bound property already set to true. You must ensure that the beforeSend request processing is idempotent, which means that it could be called multiple times and have the same effect. For important related information, see “Distributed Message Payload Transformations Before Sending” in the 9.0.1 section of the <i>New and Changed Guide</i> .
PL-35458	Fixed an issue where messaging leases are not released when changing to Maintenance run level.
Metadata Code Generator	
PL-35307	Retired typekey values now render in Java and Gosu in strikethrough text.
PCF - Layout - List View	
PLWEB-6177	Fixed an issue where clicking a check box in a list view would select more than one row.
PCF - Widget - Choice	
PLWEB-6079	Fixed an issue where radio buttons were cut off on the right side in Internet Explorer.
PCF - Widget - Range	
PLWEB-6169	Fixed an issue that caused MultiSelect range inputs to display incorrectly in certain circumstances.
PCF - Widget - Text	
PLWEB-6186	Fixed an issue where a line might wrap in the middle of a word.
Search	
PL-32634	Added Solr integration to the Management Beans page.
PL-34749	Fixed an issue that could cause an HTTP 400 error when submitting a Solr index request.
PL-35884	Fixed an issue that prevented the server locating the Solr distribution archive when server is launched from the command line and embedded Solr is enabled.
Web - PCF Compiler	
PLWEB-6088	Fixed the PCF Format Reference file so that it does not change between releases when there have been no changes to the PCF schema.
Web - Styling	
PLWEB-6049	Fixed an issue where the drop-down arrow for a disabled Tab bar menu was not disabled.
Web - UI/Runtime	
PLWEB-6165	Fixed an error that would occur when pressing Enter inside a combo box.
Web Services - WSI (New)	
PL-35045	For consuming WSDL for external web services, the product now additionally supports SOAP 1.2 schema.
XMLElement (and XSD types)	
PL-35528	For all XML code generation (XSD, GX model, WSDL), console output is now in the language specified as the default locale.

Known issues and limitations for ContactManager 9.0.1

This topic describes known issues with this release of Guidewire ContactManager.

Note: Guidewire sometimes defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 9.0.1 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not verify that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one `ContactContact` entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the `RelatedABContactSearchCriteria` entity and the `CriteriaDef` elements that are defined for it in `search-config.xml` is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

SSN / TaxID field not editable if autocomplete triggered (CTC-2989)

Issue – The **SSN/Tax ID** field of a contact is not immediately editable after an address autocomplete populates an address field.

Workaround – After autocomplete fills the address field, the **SSN/Tax ID** field has a drop-down menu button to its right. Click the button to open the menu, and choose either **Delete** or **Enter New Value**.

Studio/Platform known issues for ContactManager 9.0.1

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

Alignment Property Does Not Work for Inputs in a DetailView (PL-29429)

Issue – Inputs in a detail view do not align to the right or center. They always align to the left, regardless of the value of the `align` property.

Workaround – If you need to right align a series of items such as a set of monetary values, put them into a list view instead of a detail view. You can then set the alignment as needed.

JMS inbound integration implementations must be OSGi or Java plugins (PL-32054)

Issue – Writing JMS inbound integration handlers requires casting an object to the type `javax.jms.Message`. In Gosu, for some app servers this cast fails with the message: **Caused by:**

gw.lang.parser.exceptions.ErrantGosuClassException: GosuClass test.GosuJMSMessageReplyHandler has errors, and cannot be used at runtime.

Workaround – Implement JMS handler code for inbound integration in Java. Guidewire recommends that for inbound integration Java code, implement your code as an OSGi plugin, not a standard (non-OSGi) Java plugin.

Compilation errors after removing a delegate (PL-34619)

Issue – When you remove a delegate, some internal code is not generated properly, resulting in compilation errors.

Workaround – In Guidewire Studio, on the **Codegen** menu, click **Generate Metadata Classes**.

Free-text search is not supported on WebLogic (PL-34871)

Issue – Free-text search is not supported on a WebLogic application server.

Workaround – Guidewire is aware of this issue.

You may also need to update your ETL code to reference the new tables.

Several permissions are defined incorrectly (PL-34946)

Issue – The following permissions are defined incorrectly:

- `perm.Holiday.edit`
- `perm.Holiday.delete`
- `perm.Note.view`

Workaround – To any role that contains the `holidaymanage` permission, add the `buswkmanage` permission to that role. To any role that contains the `noteview` permission, add the `noteedit` permission to that role.

Alt+Shift+L to reload PCF files does not work when the server is run from the command line (PLWEB-5819)

Issue – If you run the server from the command line with the command `gwb runServer`, then pressing Alt+Shift+L in the application interface does not reload PCF files.

Workaround – Run the server from Guidewire Studio instead.

Server Tools downloads may time out (PLWEB-6184)

Issue – On the **Server Tools** pages, some downloads may take too long and generate a timeout error.

Workaround – Increase the value of the `WebUIAJAXTimeout` configuration parameter.

Import fails if source and target business rules data models do not match (BIZ-862)

Issue – During a business rules export and import operation from one system to another (for example, from a test system to a production system), the import fails if the business rules data model does not match between the two systems.

Workaround – Ensure that the business rules data model of the source and target systems is the same before exporting any business rule. If necessary:

1. Modify the business rules data model of one system so that it matches the business rules data model of the other system.
2. Regenerate the business rules export file.
3. Repeat the business rules import operation.

Compiling project from Studio causes rolling upgrade to fail (PL-36030)

Issue – After compiling your project from Studio, rolling upgrade reports a configuration mismatch, and that a full upgrade is required.

Workaround – Edit the file `ContactManager/modules/script/gw-build.gradle`. In that file, locate the following code block:

```
webapp {
    deployTemplates = project.rootProject.file('modules/script/deploy-templates')
    warIncludes['modules/configuration/plugins'] = file('plugins')
}
```

Replace the above code block with the following:

```
webapp {
    deployTemplates = project.rootProject.file('modules/script/deploy-templates')
    file('plugins')?.listFiles().each { File outerFile ->
        outerFile?.listFiles().each { File innerFile ->
            if(innerFile.name != 'idea-gclasses') {
                warIncludes['modules/configuration/plugins/' + outerFile.name + '/' + innerFile.name] =
innerFile
            }
        }
    }
}
```

During rule import, selecting the deployed rule results in incorrect rule version and status (BIZ-912)

Issue – Under certain circumstances, the business rule import process sets the rule version incorrectly. This occurs in the following circumstances:

1. The existing and importing rules show as a version conflict.
2. The existing or importing rule version is in the deployed state.
3. The other rule version has a status other than deployed (draft, staged, approved).
4. The deployed version of the rule is selected.

In all of these cases, the rule version is set to Approved with the deployed rule version number. For example, the deployed version is 1 and the version after import is 1+ Approved.

Workaround – Guidewire is aware of this issue.

Guidewire ContactManager 9.0.0 release notes

These release notes contain the following topics:

- “Release Information for ContactManager 9.0.0” on page 334
- “Installing ContactManager 9.0.0” on page 335
- “Support” on page 21
- “Major issues and changes for ContactManager 9.0.0” on page 336
- “Improvements and resolved issues for ContactManager 9.0.0” on page 336
- “Known issues and limitations for ContactManager 9.0.0” on page 337

Release Information for ContactManager 9.0.0

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the Release Notes for those releases to learn about changes and fixed issues. See “Release note archive” on page 301.

This release of Guidewire ContactManager is 9.0.0.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 8.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 8.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 9.0.0

For general installation information, see “Installing ContactManager” on page 55.

For versions prior to 9.0.0 that you have skipped:

- For versions subsequent to 7.0.0, see “Release note archive” on page 301.
- For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include topics for ContactCenter release notes.

Supported Java versions for ContactManager 9.0.0

The following are the supported Java versions for this release:

- Java JDK 1.8.0_92
- DCEVM version Java 8 update 92, build 1 (optional)

Version support for future releases will be published on the Guidewire Community.

Install the latest Guidewire Studio update

About this task

You can now download and apply updates to Guidewire Studio™ without needing to upgrade the entire ContactManager application. Studio automatically checks for updates and alerts you when they are available. As part of your ContactManager installation or upgrade, check for the latest update to Studio.

Procedure

1. Install Guidewire Studio following the standard instructions in the *Contact Management Guide*.
2. Run Guidewire Studio.
3. In the **Help** menu, click **Check for Studio Update**.
4. If an update is available, download and apply it.

Upgrade Information for ContactManager 9.0.0

Significant Changes to Upgrade Procedure

IMPORTANT The procedure for upgrading to ContactManager 9 from ContactManager 8 or previous releases has changed significantly. In ContactManager 9, there are new tools available for performing the upgrade. Do not rely on your past knowledge of the upgrade process to begin the upgrade procedure. Before beginning your upgrade, review the topic “Configuration Upgrade Overview” in the *Upgrade Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 9.0.0

This topic contains major issues and changes that can affect your installation.

New and changed features for ContactManager 9.0.0

For information on new features and other changes, see the topic “New and Changed Features in ContactManager 9.0.0” in the *Guidewire Contact Management Guide*.

Oracle support planned for ContactManager 9.0.0

Due to the timing of Oracle releases, this release of Guidewire InsuranceSuite 9 does not support the Oracle database. Guidewire plans to add support for Oracle, and is continuing to evaluate and test Oracle releases. For further updates about Oracle support, please contact your Guidewire account manager.

Note: In anticipation of future Oracle support, the product documentation describes the usage of Oracle with Guidewire InsuranceSuite 9. However, until Oracle is supported, this content is for informational purposes only.

Base PCF file changes for ContactManager 9.0.0

The following link requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

[ContactManager release 8.0.4 to 9.0.0](#)

To view a report of the changes to the base PCF files, refer to the original ContactManager 9.0.0 release notes.

Base rule changes for ContactManager 9.0.0

The following link requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

[ContactManager release 8.0.4 to 9.0.0](#)

To view a report of the changes to the base rules files, refer to the original ContactManager 9.0.0 release notes.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 9.0.0

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ContactManager 9.0.0 improvements and resolved issues

ID	Description
Integration-CC	
CTC-3262	An issue that prevented a ClaimCenter user with correct permissions from changing bank data (EFTData) fields for a contact stored in ContactManager has been fixed.
Miscellaneous-None	
CTC-3269	There was a problem with adding a new Person that was a vendor as a primary contact. Trying to add services in the Services tab caused an exception. This problem has been fixed.

Known issues and limitations for ContactManager 9.0.0

This topic describes known issues with this release of Guidewire ContactManager.

Note: Guidewire sometimes defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 9.0.0 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Still an issue in 9.0.0

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not verify that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one `ContactContact` entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Still an issue in 9.0.0

Issue – Using the `RelatedABContactSearchCriteria` entity and the `CriteriaDef` elements that are defined for it in `search-config.xml` is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

SSN / TaxID field not editable if autocomplete triggered (CTC-2989)

Still an issue in 9.0.0

Issue – The **SSN/Tax ID** field of a contact is not immediately editable after an address autocomplete populates an address field.

Workaround – After autocomplete fills the address field, the **SSN/Tax ID** field has a drop-down menu button to its right. Click the button to open the menu, and choose either **Delete** or **Enter New Value**.

Studio/Platform known issues for ContactManager 9.0.0

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

Alignment Property Does Not Work for Inputs in a DetailView (PL-29429)

Issue – Inputs in a detail view do not align to the right or center. They always align to the left, regardless of the value of the `align` property.

Workaround – If you need to right align a series of items such as a set of monetary values, put them into a list view instead of a detail view. You can then set the alignment as needed.

JMS inbound integration implementations must be OSGi or Java plugins (PL-32054)

Issue – Writing JMS inbound integration handlers requires casting an object to the type `javax.jms.Message`. In Gosu, for some app servers this fails with the message: “Caused by: gw.lang.parser.exceptions.ErrantGosuClassException: GosuClass test.GosuJMSMessageReplyHandler has errors, and cannot be used at runtime.”

Workaround – Implement JMS handler code for inbound integration in Java. Guidewire recommends that for inbound integration Java code, implement your code as an OSGi plugin, not a standard (non-OSGi) Java plugin.

Compilation errors after removing a delegate (PL-34619)

Issue – When you remove a delegate, some internal code is not generated properly, resulting in compilation errors.

Workaround – In Guidewire Studio, on the **Codegen** menu, click **Generate Metadata Classes**.

Free-text search is not supported on WebLogic (PL-34871)

Issue – Free-text search is not supported on a WebLogic application server.

Workaround – Guidewire is aware of this issue.

You may also need to update your ETL code to reference the new tables.

Several permissions are defined incorrectly (PL-34946)

Issue – The following permissions are defined incorrectly:

- `perm.Holiday.edit`
- `perm.Holiday.delete`
- `perm.Note.view`

Workaround – To any role that contains the `holidaymanage` permission, add the `buswkmanage` permission to that role. To any role that contains the `noteview` permission, add the `noteedit` permission to that role.

Alt+Shift+L to reload PCF files does not work when the server is run from the command line (PLWEB-5819)

Issue – If you run the server from the command line with the command `gwb runServer`, then pressing `Alt+Shift+L` in the application interface does not reload PCF files.

Workaround – Run the server from Guidewire Studio instead.

Server Tools downloads may time out (PLWEB-6184)

Issue – On the **Server Tools** pages, some downloads may take too long and generate a timeout error.

Workaround – Increase the value of the `WebUIAJAXTimeout` configuration parameter.

Parenthesis and RuleOperator typelists are not localizable (BIZ-822)

Issue – The Parenthesis and RuleOperator typelists are not localizable, and any localization causes Gosu compilation to fail.

Workaround – Guidewire is aware of this issue.

Import fails if source and target business rules data models do not match (BIZ-862)

Issue – During a business rules export and import operation from one system to another (for example, from a test system to a production system), the import fails if the business rules data model does not match between the two systems.

Workaround – Ensure that the business rules data model of the source and target systems is the same before exporting any business rule. If necessary:

1. Modify the business rules data model of one system so that it matches the business rules data model of the other system.
2. Regenerate the business rules export file.
3. Repeat the business rules import operation.

Guidewire ContactManager 8.0.7 release notes

These release notes contain the following topics:

- “Release Information for ContactManager 8.0.7” on page 339
- “Installing ContactManager 8.0.7” on page 339
- “Support” on page 21
- “Major issues and changes for ContactManager 8.0.7” on page 340
- “Improvements and resolved issues for ContactManager 8.0.6” on page 346
- “Known issues and limitations for ContactManager 8.0.6” on page 349

Release Information for ContactManager 8.0.7

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the Release Notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 8.0.7” on page 339.

This release of Guidewire ContactManager is 8.0.7.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 7.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 8.0.7

For general installation information, see “Installing ContactManager” on page 55.

For versions of ContactManager prior to 8.0.7 that you have skipped, see:

- “Guidewire ContactManager 8.0.6 release notes” on page 344
- “Guidewire ContactManager 8.0.5 release notes” on page 352
- “Guidewire ContactManager 8.0.4 release notes” on page 359
- “Guidewire ContactManager 8.0.3 release notes” on page 366
- “Guidewire ContactManager 8.0.2 release notes” on page 377
- “Guidewire ContactManager 8.0.1 release notes” on page 388
- “Guidewire ContactManager 8.0.0 release notes” on page 400
- “Guidewire ContactManager 7.0.7 release notes” on page 405
- “Guidewire ContactManager 7.0.6 release notes” on page 412
- “Guidewire ContactManager 7.0.5 release notes” on page 416
- “Guidewire ContactManager 7.0.4 release notes” on page 423
- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

Upgrade Information for ContactManager 8.0.7

Incorrect warning message when running Configuration Upgrade Tool (PL-28723)

When upgrading from one ContactManager 8.0.x release to another ContactManager 8.0.x release, running the Configuration Upgrade Tool may produce following warning message:

WARN cannot find Emerald base configuration zip, this could indicate a problem...

If the ContactManager/modules/base.zip file does exist, then you can safely ignore this warning.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 8.0.7

These release notes list major issues and changes that can affect your installation.

New and changed features for ContactManager 8.0.7

The primary new feature in this release is Personal Data Destruction. This feature enables you to comply with some of your personal data destruction requirements.

For information on new features and other changes, see “New and changed features in ContactManager 8.0.7” on page 35.

Base PCF file changes for ContactManager 8.0.7

ContactManager release 8.0.6 to 8.0.7

To view a report of the changes to the base PCF files, refer to the original ContactManager 8.0.7 release notes.

Base rule changes for ContactManager 8.0.7

There are no changes to rules in this release.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the [Guidewire Community](#).

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 8.0.7

There are improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. The list in these release notes is not intended to be comprehensive.

Platform improvements, resolved issues for ContactManager 8.0.7

ID	Description
Database Instrumentation	
PL-36680	Correctly ordered the column headings in the Message Queue Depth report in the AWR download on the Server Tools page.
Database Upgrade	
PL-37095	Fixed an issue that caused upgrade to fail if you modified <code>GWNormalize</code> in <code>collations.xml</code> , and it did not mimic the semantics of the <code>LOWER</code> database function.
Entities/Metadata	
PL-37291	Fixed an issue that could cause an exception if bundle contents changed during iteration of the beans in the bundle. The <code>getBeansByRootType</code> method now returns a collection that is an immutable copy of the bean values in the bundle.
Inbound Integration	
PL-37136	Fixed an issue in JMS inbound integration with the ordered parameter set to <code>false</code> where polling an empty thread did not close the JMS session.
PL-37293	Fixed an issue that caused the inbound file integration to stop processing files if the incoming folder contained more than 300 files.
PCF - Layout - List View - Pagination	
PLWEB-6645	Fixed an issue where an editable list view would not display data when you used the pagination controls to move through its multiple pages.
PCF - Layout - Workspace and Worksheets	
PLWEB-4571	Fixed an issue where validation messages were not visible when they were displayed in a Workspace panel that was hidden. If the Workspace is hidden when new messages appear, the Workspace now opens automatically.
PCF - Widget - Cell (Generic)	
PLWEB-6642	Fixed an issue where the <code>wrap</code> attribute on a <code>Cell</code> was not respected by <code>FormalCell</code> or <code>LinkCell</code> .
PCF - Widget - Toolbar	
PLWEB-3303	Added the <code>reflectOnBottom</code> attribute to the PCF Toolbar widget. Setting <code>reflectOnBottom</code> to <code>true</code> causes the toolbar to appear both at the top and bottom of the screen.

ID	Description
Web - UI/Runtime	
PLWEB-1754	Fixed an issue with validation messages not having the correct page label.

Known issues and limitations for ContactManager 8.0.7

This topic describes known issues with this release of Guidewire ContactManager.

Note: For maintenance releases, Guidewire often defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 8.0.7 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one ContactContact entity. ContactManager does not verify that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant ContactContact entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one ContactContact entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the RelatedABContactSearchCriteria entity and the CriteriaDef elements that are defined for it in search-config.xml is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

Contact mapping code requires initially capitalized array extension names (CTC-4152)

Issue – In ContactIntegrationXLMapperAppBase#populateArrayBean, the code uses the array name to call the addTo and removeFrom methods to add or remove elements from the array. However the code generator creates methods with the initial character of the property in upper case. If the property name starts with a lowercase letter, calling one of these generated methods causes a Property Not Found error. For example, for an array property ex_ContactDrpServices, the generated methods addToEx_ContactDrpServices and removeFromEx_ContactDrpServices do not find the property, and invoking them fails.

Workaround – If you add an array extension to an ABContact entity or one of its subentities, use an uppercase letter for the first letter in the name.

If you have already created array extensions with names that start with a lowercase letter, you can find them and correct the names as follows:

1. In Guidewire Studio™, open the ContactManager ContactMapper class.
2. Search for all occurrences of arrayMapping.
3. For each occurrence, check if the mapping is of an array with a lowercase first letter. If so, make a record of it so you can fix it in code. For example, lowercase Arrays Found:

```
ex_ContactDrpHours, ex_ContactDrpPayments, ex_ContactDrpServices
```

4. For each lowercase array name found, go to the entity where it is defined and change the first letter to a capital. For example:

```
Ex_ContactDrpHours, Ex_ContactDrpPayments, Ex_ContactDrpServices
```

5. You must also change each matching array extension in the core applications that use ContactManager.

Studio/Platform known issues for ContactManager 8.0.7

Long text in table cells can add white space to the right of the page (PL-28288)

Issue – In an editable list view, extremely long text entered in a single cell can cause additional white space on the right side of the page. Long text is text that occupies approximately the entire width of the screen. This issue occurs primarily in Chrome.

Workaround – If you expect users to enter large amounts of text into the cells of a column, configure the column to support text wrapping.

Administrative command-line tools cannot refresh WSDL (PL-29021)

Issue – Administrative command-line tools rely on web service implementation classes such as `MaintenanceToolsAPI.gs`. If you change the web service implementation classes, administrative tools might fail because the tool's WSDL does not match the server WSDL. Some changes do not affect the WSDL. For example, adding a `@WsiPermission` annotation.

Workaround – Do the following:

From the `ContactManager/bin` directory, at a command prompt type the command:

```
gwab regen-soap-api
```

In Windows Explorer, copy the WSDL files from the location:

```
ContactManager/soap-api/wsi/wsd1
```

to:

```
ContactManager/admin/res/wsi
```

No entity changed event fired through a one-to-one entity relationship (PL-26224)

Issue – The existence of a one-to-one entity relationship between two entities prevents an entity change event from being fired, even if it is supposed to. For example, in ContactManager there is a foreign key from `ABContact` to `Address` for the primary address, and a one-to-one relationship from `Address` to `ABContact`. Because of the one-to-one link, no event changed event is fired for any change in the primary address.

More generally, if entity E1 has a one-to-one relationship to entity E2, no entity changed event is fired for E2 if a property on E1 changes.

Workaround – You might be able to take advantage of a secondary change caused by the initial change, with the secondary change triggering the event to be fired instead. In the previous ContactManager example, making sure that changes to all properties in `Address` cause a `History` record to be generated ensures that the entity changed event will be fired.

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

Alignment property does not work for inputs in a detail view (PL-29429)

Issue – Inputs in a detail view do not align to the right or center. They always align to the left, regardless of the value of the `align` property.

Workaround – If you need to right align a series of items such as a set of monetary values, put them into a list view instead of a detail view. You can then set the alignment as needed.

JMS inbound integration implementations must be OSGi or Java plugins (PL-32054)

Issue – Writing JMS inbound integration handlers requires casting an object to the type `javax.jms.Message`. In Gosu, for some app servers this fails with the message: `Caused by: gw.lang.parser.exceptions.ErrantGosuClassException: GosuClass test.GosuJMSMessageReplyHandler has errors, and cannot be used at runtime.`

Workaround – Implement JMS handler code for inbound integration in Java. Guidewire recommends that for inbound integration Java code, implement your code as an OSGi plugin, not a standard (non-OSGi) Java plugin.

Asynchronous pending documents when IDocumentMetadataSource plugin is disabled (PL-34975)

Issue – There is a known issue if you use asynchronous document content storage and disable the `IDocumentMetadataSource` plugin, which is the default configuration. With the `IDocumentMetadataSource` plugin disabled, the application stores document metadata in the local database. A recently created document can be pending, which means the `Document` entity instance is in the internal database, but its contents are not yet sent to the DMS. Pending documents appear in document lists, but viewing or editing the document fails. Also, if you rename a document while it is pending, the file is orphaned and never uploaded to the DMS.

Workaround – Guidewire is aware of this issue.

InputHelpTextOnFocus parameter is no longer supported (PLWEB-5852)

Issue – The configuration parameter `InputHelpTextOnFocus` is no longer supported.

Workaround – Guidewire is aware of this issue.

Guidewire ContactManager 8.0.6 release notes

These release notes contain the following topics:

- “Release Information for ContactManager 8.0.6” on page 344
- “Installing ContactManager 8.0.6” on page 345
- “Support” on page 21
- “Major issues and changes for ContactManager 8.0.6” on page 345
- “Improvements and resolved issues for ContactManager 8.0.6” on page 346
- “Known issues and limitations for ContactManager 8.0.6” on page 349

Release Information for ContactManager 8.0.6

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the Release Notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 8.0.6” on page 345.

This release of Guidewire ContactManager is 8.0.6.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 7.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 8.0.6

For general installation information, see “Installing ContactManager” on page 55.

For versions of ContactManager prior to 8.0.6 that you have skipped, see:

- “Guidewire ContactManager 8.0.5 release notes” on page 352
- “Guidewire ContactManager 8.0.4 release notes” on page 359
- “Guidewire ContactManager 8.0.3 release notes” on page 366
- “Guidewire ContactManager 8.0.2 release notes” on page 377
- “Guidewire ContactManager 8.0.1 release notes” on page 388
- “Guidewire ContactManager 8.0.0 release notes” on page 400
- “Guidewire ContactManager 7.0.7 release notes” on page 405
- “Guidewire ContactManager 7.0.6 release notes” on page 412
- “Guidewire ContactManager 7.0.5 release notes” on page 416
- “Guidewire ContactManager 7.0.4 release notes” on page 423
- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

Upgrade Information for ContactManager 8.0.6

Incorrect warning message when running Configuration Upgrade Tool (PL-28723)

When upgrading from one ContactManager 8.0.x release to another ContactManager 8.0.x release, running the Configuration Upgrade Tool may produce following warning message:

WARN cannot find Emerald base configuration zip, this could indicate a problem...

If the ContactManager/modules/base.zip file does exist, then you can safely ignore this warning.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 8.0.6

These release notes list major issues and changes that can affect your installation.

New and changed features for ContactManager 8.0.6

There is one major changed feature in ContactManager 8.0.6, required changes to geocoding with Bing Maps.

See also

- “New and changed features in ContactManager 8.0.6” on page 35

Base PCF file changes for ContactManager 8.0.6

ContactManager release 8.0.5 to 8.0.6

To view a report of the changes to the base PCF files, refer to the original ContactManager 8.0.6 release notes.

Base rule changes for ContactManager 8.0.6

To view a report of the changes to the base rule files, refer to the original ContactManager 8.0.6 release notes.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 8.0.6

There are improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. The list in these release notes is not intended to be comprehensive.

ContactManager 8.0.6 improvements and resolved issues

ID	Description
Contact Change Management	
CTC-3899	Contact history records now display the username rather than the display name for a given contact change. It was not always possible to show the display name, so contact history is now more consistent.
Merge Contacts	
CTC-3950	Fixed an issue with updating of local copies of contacts in ClaimCenter after a merge in ContactManager. All local copies of contacts in ClaimCenter are now getting updated with the latest contact data after contacts are merged in ContactManager.

Platform improvements, resolved issues for ContactManager 8.0.6

ID	Description
Clustering	
PL-35116	The serialization whitelist (serialization-whitelist.lst) now correctly contains the following classes: <ul style="list-style-type: none">• ConfigVerificationResults• ConfigVerificationResults\$FileMismatchInfo
Cognos Integration	

ID	Description
PL-34886	Fixed an issue that caused the Cognos plugin to not start on non-batch server nodes, causing the Report tab to not be visible.
Contact Domain	
PL-34674	Fixed an issue in which duplicate insertions of <code>ContactFingerprint</code> entities could cause a database exception.
Database Instrumentation	
PL-34649	When running against an Oracle database, the Database Table Info download now includes information on all indexes that have been marked <code>INVISIBLE</code> .
PL-34656	When running against an Oracle database, the Database Table Info download now includes information on any tables or indexes with object-level parallelism.
PL-35172	The Server Tools Database Table Info report now includes information on the interfaces implemented by the entity underlying table (or the entity subtype, if the entity is subtyped). To access the information, click Implemented Interfaces in the Database Table info index.
PL-36364	Fixed an intermittent issue with Oracle AWR reports caused by the use of an incorrect datatype
Database Support - General	
PL-34587	<p><code>connections-initialized-for-application</code> is a new attribute that you can add to the <code>jndi-connection-pool</code> element in <code>database-config.xml</code>. This attribute only affects JNDI database connections on Oracle databases. Prior to this change, every time the database service borrowed a connection from an external data source, it updated the values of certain database parameters. Although the cost of executing these updates is very small, Guidewire applications sometimes borrow connections at a rapid rate, and the time to execute these updates can sometimes add up and become visible in performance analyses.</p> <p>The attribute takes one of the following values:</p> <ul style="list-style-type: none"> <code>true</code> -- The database service checks the following database parameters at server startup: <code>CURSOR_SHARING</code>, <code>MODULE</code>, <code>NLS_SORT</code>, and <code>NLS_COMP</code>. If the values are set correctly, the database service does not update the database parameters thereafter. If the values are not set correctly, the application server stops and prints an informative error message. <code>false</code> -- The database service performs the updates every time that it establishes a new JNDI connection to the database. <p>To use this setting, apply it to the existing JNDI database connection. Any generated error message contains information on which settings are correct and which ones still need to be applied.</p>
PL-34597	Fixed an issue with high CPU usage on the database server while running custom batch processes.
Database Support - Oracle	
PL-34974	Fixed an issue where the parallel degree for newly created indexes on primary keys was not being reset after a database upgrade.
Database Support - SQL Server	
PL-36224	Fixed an issue where upgrading a SQL Server database from 7.0 to 8.0 caused a data truncation error.
Document Management	
PL-36617	Fixed an issue where Note links to a document were broken if the <code>IDocumentMetadataSource</code> plugin was enabled.
Email	
PL-34088	Fixed a <code>NullPointerException</code> issue when processing an email in the <code>MessageTransport</code> plugin. The issue occurred whenever the recipient's email address was invalid.

ID	Description
Entities/Metadata	
PL-36265	Fixed an issue that caused the <code>regen-java-api</code> command to produce a compilation error whenever <code>IMSignPart</code> was extended to implement <code>Coverable</code> .
GX Tools	
PL-36756	To reduce the size of the GX model heap space, Guidewire recommends turning off the validation check that occurs when writing GX models.
Gosu	
PL-34541	Improved the performance for queries returning a large number of results.
Inbound Integration	
PL-36631	Fixed an issue where file-based inbound integration did not process files that existed in the incoming directory at server startup.
Infrastructure	
PL-31383	Fixed an issue where the PolicyCenter ready message appeared before the user could log in.
Integration	
PL-33556	Fixed an issue where the <code>suite-config.xml</code> file's <code>gw.ab.env</code> parameter would be ignored.
Messaging	
PL-33317	Fixed an issue where paging was not enabled to scroll through the entries on the Administration→Monitoring→Message Queues→Destination screen.
PL-35160	To improve troubleshooting operations, all failures to suspend or resume a message destination are logged.
Other	
PLWEB-6159	When an exception occurs while rendering the UI, the error log now records the source location of the exception.
PLWEB-6201	Fixed an issue where putting the server into maintenance mode would not show a message in the UI stating that the server is undergoing maintenance.
PLWEB-6285	Replaced several hard-coded strings in JavaScript files with display keys.
PCF - Actions - Auto Complete	
PLWEB-2033	Fixed an issue where you could not type into a TaxID or SSN text box after navigating from a field that used auto-completion.
PCF - Layout - List View	
PLWEB-4287	In a list view, a <code>TextCell</code> with an input mask set is no longer cleared when it contains an invalid value and you tab out of it.
PLWEB-4436	Fixed an issue where the cursor focus on a page would skip cells in row iterator widgets.
PCF - Navigation - Tab Bar - Quick Jump	
PLWEB-5029	Fixed an issue where the suggested commands in the QuickJump box were not always correct for the application context.
PCF - Widget - Address AutoFill	
PLWEB-6057	Fixed an issue where an invalid postal code would sometimes not be highlighted.
PCF - Widget - Modal Cell	
PLWEB-5986	Fixed an issue where text would not wrap in a modal cell in a list view.
PCF - Widget - Text	

ID	Description
PLWEB-6191	Fixed an issue with line wrapping in list views.
Persistence	
PL-34926	Fixed a <code>ConcurrentDataChangeException</code> when using WebLogic and Oracle with JNDI.
Search	
PL-34739	Fixed a problem that passed invalid command line options to external sort on AIX during Solr batch load.
PL-34798	Fixed a problem where loading data that contained the character sequence “###” caused Solr batch load to fail.
Web - Configuration	
PLWEB-6303	Fixed an issue where a list view with <code>numEntriesRequired</code> specified might incorrectly produce a validation error stating that the required number of rows has not been met.
Web - Showcase	
PLWEB-6299	Fixed an issue where a <code>BooleanRadioInput</code> that toggled the visibility of other screen elements would cause excessive screen flickering.
Web - UI/Runtime	
PLWEB-2022	Fixed an issue where user interface shortcut keys would not work.
PLWEB-5455	Fixed an error that would occur when pressing Enter inside a combo box.
PLWEB-5616	Fixed an issue where selecting multiple rows in two separate list views and then clicking Remove would remove the rows from both list views. Instead, only the rows in the list view containing the clicked button are removed.
PLWEB-5767	Fixed an issue where clicking on field when an off-screen field had focus would cause the screen to jump to the field with previous focus.
Web Services - RPCE (Old)	
PL-35522	Fixed an issue that created a large number of HTTP sessions when using RPC web services.
Web Services - WSI (New)	
PL-35158	Fixed an issue when using web services collections that resulted in the default web service URL being selected rather than a specified override URL.
PL-36749	Fixed an issue where a SOAP MTOM (Message Transmission Optimization Mechanism) response that contained multiple sequential CR-LF pairs would throw an exception.
XMLElement (and XSD types)	
PL-32120	Fixed an issue that truncated trailing zeros from the milliseconds portion of a datetime field in a GX model.
PL-34746	Performance has been improved when parsing XML. ♦

Known issues and limitations for ContactManager 8.0.6

This topic describes known issues with this release of Guidewire ContactManager.

Note: For maintenance releases, Guidewire often defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 8.0.6 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one ContactContact entity. ContactManager does not verify that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant ContactContact entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one ContactContact entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the RelatedABContactSearchCriteria entity and the CriteriaDef elements that are defined for it in search-config.xml is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

Studio/Platform known issues for ContactManager 8.0.6

Long text in table cells can add white space to the right of the page (PL-28288)

Issue – In an editable list view, extremely long text entered in a single cell can cause additional white space on the right side of the page. Long text is text that occupies approximately the entire width of the screen. This issue occurs primarily in Chrome.

Workaround – If you expect users to enter large amounts of text into the cells of a column, configure the column to support text wrapping.

Administrative command-line tools cannot refresh WSDL (PL-29021)

Issue – Administrative command-line tools rely on web service implementation classes such as MaintenanceToolsAPI.gs. If you change the web service implementation classes, administrative tools might fail because the tool's WSDL does not match the server WSDL. Some changes do not affect the WSDL. For example, adding a @WsiPermission annotation.

Workaround – Do the following:

From the ContactManager/bin directory, at a command prompt type the command:

```
gwab regen-soap-api
```

In Windows Explorer, copy the WSDL files from the location:

```
ContactManager/soap-api/wsi/wsd1
```

to:

```
ContactManager/admin/res/wsi
```

No entity changed event fired through a one-to-one entity relationship (PL-26224)

Issue – The existence of a one-to-one entity relationship between two entities prevents an entity change event from being fired, even if it is supposed to. For example, in ContactManager there is a foreign key from ABContact to Address for the primary address, and a one-to-one relationship from Address to ABContact. Because of the one-to-one link, no event changed event is fired for any change in the primary address.

More generally, if entity E1 has a one-to-one relationship to entity E2, no entity changed event is fired for E2 if a property on E1 changes.

Workaround – You might be able to take advantage of a secondary change caused by the initial change, with the secondary change triggering the event to be fired instead. In the previous ContactManager example, making sure that changes to all properties in Address cause a History record to be generated ensures that the entity changed event will be fired.

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

Alignment property does not work for inputs in a detail view (PL-29429)

Issue – Inputs in a detail view do not align to the right or center. They always align to the left, regardless of the value of the `align` property.

Workaround – If you need to right align a series of items such as a set of monetary values, put them into a list view instead of a detail view. You can then set the alignment as needed.

JMS inbound integration implementations must be OSGi or Java plugins (PL-32054)

Issue – Writing JMS inbound integration handlers requires casting an object to the type `javax.jms.Message`. In Gosu, for some app servers this fails with the message: `Caused by: gw.lang.parser.exceptions.ErrantGosuClassException: GosuClass test.GosuJMSMessageReplyHandler has errors, and cannot be used at runtime.`

Workaround – Implement JMS handler code for inbound integration in Java. Guidewire recommends that for inbound integration Java code, implement your code as an OSGi plugin, not a standard (non-OSGi) Java plugin.

Asynchronous pending documents when IDocumentMetadataSource plugin is disabled (PL-34975)

Issue – There is a known issue if you use asynchronous document content storage and disable the `IDocumentMetadataSource` plugin, which is the default configuration. With the `IDocumentMetadataSource` plugin disabled, the application stores document metadata in the local database. A recently created document can be pending, which means the `Document` entity instance is in the internal database, but its contents are not yet sent to the DMS. Pending documents appear in document lists, but viewing or editing the document fails. Also, if you rename a document while it is pending, the file is orphaned and never uploaded to the DMS.

Workaround – Guidewire is aware of this issue.

InputHelpTextOnFocus parameter is no longer supported (PLWEB-5852)

Issue – The configuration parameter `InputHelpTextOnFocus` is no longer supported.

Workaround – Guidewire is aware of this issue.

Guidewire ContactManager 8.0.5 release notes

These release notes contain the following topics:

- “Release information for ContactManager 8.0.5” on page 352
- “Installing ContactManager 8.0.5” on page 352
- “Support” on page 21
- “Major issues and changes for ContactManager 8.0.5” on page 353
- “Improvements and resolved issues for ContactManager 8.0.5” on page 353
- “Known issues and limitations for ContactManager 8.0.5” on page 357

Release information for ContactManager 8.0.5

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the Release Notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 8.0.5” on page 352

This release of Guidewire ContactManager is 8.0.5.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 7.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 8.0.5

For general installation information, see “Installing ContactManager” on page 55.

For versions of ContactManager prior to 8.0.5 that you have skipped, see:

- “Guidewire ContactManager 8.0.4 release notes” on page 359
- “Guidewire ContactManager 8.0.3 release notes” on page 366
- “Guidewire ContactManager 8.0.2 release notes” on page 377
- “Guidewire ContactManager 8.0.1 release notes” on page 388
- “Guidewire ContactManager 8.0.0 release notes” on page 400
- “Guidewire ContactManager 7.0.7 release notes” on page 405
- “Guidewire ContactManager 7.0.6 release notes” on page 412
- “Guidewire ContactManager 7.0.5 release notes” on page 416
- “Guidewire ContactManager 7.0.4 release notes” on page 423
- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

Upgrade information for ContactManager 8.0.5

Incorrect warning message when running Configuration Upgrade Tool (PL-28723)

When upgrading from one ContactManager 8.0.x release to another ContactManager 8.0.x release, running the Configuration Upgrade Tool may produce following warning message:

`WARN cannot find Emerald base configuration zip, this could indicate a problem...`

If the `ContactManager/modules/base.zip` file does exist, then you can safely ignore this warning.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 8.0.5

These release notes list major issues and changes that can affect your installation.

New and changed features for ContactManager 8.0.5

There are no new or changed features in ContactManager 8.0.5.

Base PCF file changes for ContactManager 8.0.5

ContactManager release 8.0.4 to 8.0.5

To view a report of the changes to the base PCF files, refer to the original ContactManager 8.0.5 release notes.

Base rule changes for ContactManager 8.0.5

There were no changes to the rules between ContactManager 8.0.4 and 8.0.5.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 8.0.5

There are improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. The list in these release notes is not intended to be comprehensive.

ContactManager 8.0.5 improvements and resolved issues

ID	Description
Core	
CTC-2637	There was an issue with ContactManager reusing addresses after a contact merge, sometimes resulting in sharing of addresses by multiple contacts. Contacts must not share address objects. ContactManager now generates a copy of each address that a user who is merging contacts adds from the retired contact to the kept contact.
CTC-3497	Fixed a pagination issue with Microsoft SQL Server, which was throwing an exception for a proximity search that produced multiple pages of results.
Database	

CTC-3534	Fixed a problem with code that collects geocoding information in the data distribution batch process if the Geocoding plugin is enabled. The batch process no longer throws an exception when the Geocoding plugin is enabled. The data distribution batch process creates a report describing the distribution of data in the database, such as how many claims are in the database, the number of exposures and contacts per claim, and so on.
----------	--

Integration-CC

CTC-3550	When a pending change is approved for contact data that is not a field on the contact entity, such as an address change, ContactManager now notifies the core application that the contact has changed.
----------	---

Platform improvements and resolved issues for ContactManager 8.0.5

ID	Description
Appserver Support - WebLogic	
PL-33301	Resolved an issue that caused the Guidewire Profiler to not show operation names for RPC Web Services.
Archiving	
PL-32879	Resolved an issue with restoring entities in archived documents that contain cross-graph links in the archiving domain graph.
Clustering	
PL-34221	Improved the security of Guidewire cluster operations using JGroups by minimizing the possibility of deserialized Java objects being used to perform remote code invocation attacks against a Guidewire application server. By default, Guidewire applications disallow any Java classes placed on a blacklist to be deserialized. If desired, use the <code>SerializationWhitelistEnabled</code> configuration parameter to place certain Java classes on a whitelist to be deserialized. Server Tools Serialization Info (under Info Pages) provides a means of monitoring the deserialization of Java classes.
Consistency Checker	
PL-33402	The Consistency Checks Info Page now provides a Download Errors button to download only the checks that had errors. This type of download eliminates reporting on the checks that succeeded, thus, speeding up the download process.
Database Instrumentation	
PL-32449	Fixed several issues that caused Null Pointer exceptions while downloading the Server Tools Oracle AWR reports.
PL-33517	Fixed an issue with the Oracle Statspack download where it did not show queries under various categories.
Database Support - General	
PL-31932	To speed up staging table loading, Guidewire has added the ability to drop deferrable indexes before the insert operation and recreate them afterwards.
Database Support - Oracle	
PL-30941	Guidewire applications now use Oracle JDBC driver 12.1.0.2.0, which fixes several Protocol violation exceptions.
Database Support - SQL Server	
PL-32898	Improved the performance of the production of the (Server Tools Info Pages) SQL Server DMV Snapshot by restricting the data extract to the top 400 queries.
Database Upgrade	
PL-22087	You can now change the precision and scale of a numeric column in the data model configuration and the server will update the database next time it starts. You must ensure that digits to the left and right of the decimal point are not lost due to the change.
Entities/Metadata	

ID	Description
PL-29536	Resolved an issue where GUnit tests that created an entity would fail because no implicit bundle was passed in.
PL-29955	Resolved an issue where a typekey with the same name as another retired typekey would sort incorrectly in a typelist.
PL-34196	Fixed a performance issue with DistributedKeyGenerator.getNextID.
Gosu	
PL-32552	Resolved an issue in Gosu in which, for rare cases, an overridden method is called on the superclass not the subclass.
Inbound Integration	
PL-33163	Resolved an issue that resulted in an error when changing server run level from maintenance to multiuser.
Logging	
PLWEB-5876	To improve the tracking and debugging of CSRF token not found in the request error messages in the application log, the error message now contains session and user information.
Messaging	
PL-33144	To improve messaging database query performance on Oracle, messaging queries now limit result set size with a rownum clause.
Other	
PLWEB-5260	Resolved a cross-site scripting security issue in the scroll position.
PLWEB-5360	Added more explicit documentation about the Terms of Service to the customer.js file. Note that the modification or additional use of certain JavaScript APIs in the product is not permitted.
Other - Cloud	
PL-34270	Upgraded to a release of the Apache Commons Collections library that fixes a security vulnerability.
PCF - Layout - List View	
PLWEB-1774	Resolved an issue where using the arrow keys to scroll down in a wide list view would show excessive screen flickering.
PLWEB-4618	Changed table cell selection to not persist between page views.
PLWEB-5673	Resolved an issue where columns that were hidden would reappear after the page changed to Edit mode.
PLWEB-5715	Fixed an issue where if you have a List Detail Panel showing the details of a newly created bean in a List View, and then you click Cancel , the exception <code>IllegalStateException: Attempt to access bean with null bundle</code> would be thrown.
PCF - Layout - Workspace and Worksheets	
PLWEB-4305	Resolved an issue where triggering an ExitPoint would lead to pending updates on worksheets—such as changes to user notes—being lost.
PCF - Navigation - Left Navigation Panel	
PLWEB-5393	Resolved an issue where the highlight in the Action menu would display incorrectly.
PCF - Navigation - Tab Bar - Quick Jump	
PLWEB-4431	Resolved an issue where incorrect text was displayed in the QuickJump box while typing a command.
Plugins	
PL-33095	Resolved <code>IllegalArgumentException</code> thrown by the <code>GosuPluginContainer.getResourceAsStream</code> method.
Profiling	

ID	Description
PL-33521	Improved the profiling results of the (Server Tools) Guidewire Profiler. The Profiler now merges data collected on background threads (such as message destination sender threads) into the overall visible profiling results.
Queries	
PL-34312	The query builder API no longer generates table aliases for subselects that exceed the database limit for name length.
Security	
PL-33622	Upgraded Apache WSS4J to version 1.6.18.
Staging Tables	
PL-32354	(Oracle) To improve performance, the <code>-integritycheckandload</code> option of the <code>tableimport</code> command now calculates row counts on production tables only if you specify the <code>-estimateorastats</code> command option. Otherwise, the <code>-integritycheckandload</code> option uses information in database statistics to calculate approximate row counts. Use the <code>-estimateorastats</code> option to load production tables that are empty or have very few rows.
PL-32546	Back-to-back integrity check and load operations now fail before actually running if the first loader fails during insert/select.
Web - Configuration	
PLWEB-2083	Restored the functionality of IME mode in Internet Explorer. Note that this mode is not available in other browsers.
Web - IE Support	
PLWEB-4198	Resolved an issue in PCF elements that have the attribute <code>download</code> set to <code>true</code> . When these attributes triggered an error in Internet Explorer, the browser would prompt to download a <code>.do</code> or <code>.json</code> file instead of displaying an error message. Now all browsers display the proper error message.
Web - Performance	
PLWEB-5278	Resolved a performance issue where searches with no returned values ran unnecessary, additional queries.
Web - UI/Runtime	
PLWEB-1788	Addressed an issue where certain ExitPoint windows, such as the About or Help windows, were hidden behind the main browser window in Internet Explorer. This issue is partially due to Internet Explorer behavior, but this release minimizes the likelihood that it occurs.
PLWEB-1922	Resolved an issue where the progress bar was not displaying correctly.
PLWEB-2093	Added a <code>pickWidth</code> property to the Select column in tables. This allows a wider column to support configurations with longer alternatives to the word "Select".
PLWEB-2123	Resolved an issue where the progress bar would display "-1%" instead of an accurate progress percentage.
PLWEB-4290	Improved the error message if the web UI times out when trying to contact the server. Also added the new configuration parameter <code>WebUIAJAXTimeout</code> , which specifies the number of seconds to wait before showing a timeout message. The default value is 600 (10 minutes).
PLWEB-4304	Improved and resolved issues with keyboard navigation through the Actions menu.
PLWEB-4554	Resolved an issue where the JavaScript action wasn't working for a <code>menuItem</code> .
PLWEB-4645	Resolved an issue where <code>textField</code> and <code>typekey</code> cells in a <code>ListView</code> could show the wrong display name.
PLWEB-4690	Improved the security of the CSRF token by removing it from URLs.
PLWEB-5510	Resolved an issue where half-width characters in Japanese were not wrapping in Google Chrome.
PLWEB-5580	Resolved an issue with the Bing Maps implementation and its use of the CSRF token.

ID	Description
PLWEB-5616	Fixed an issue where selecting multiple rows in two separate List Views and then clicking Remove would remove the rows from both List Views. Instead, only the rows in the List View containing the clicked button are removed.
PLWEB-5953	Fixed an issue where a JavaScript error occurred on the Japanese Imperial Date widget when the Regional Format was changed.
Web Services - WSI (New)	
PL-34336	Introduced the new annotation <code>@WsiReduceDBConnections</code> , which limits the number of connection requests that a web service call makes to the database connection pool. This annotation is not generally needed, and using it might cause a faulty or slow external web service to keep database connections active and use up the connection pool. Use this annotation with caution, as it defeats the purpose of the connection pool.
XMLElement (and XSD types)	
PL-31865	Resolved compatibility issue with WSS4J and xmlsec Java libraries.
PL-32193	Improved XML sorting performance during serialization.
PL-33338	For improved security, XML documents cannot use DTDs. This affects XML parsed by WS-I web services or XSDs.

Known issues and limitations for ContactManager 8.0.5

This topic describes known issues with this release of Guidewire ContactManager.

Note: For maintenance releases, Guidewire often defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 8.0.5 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not verify that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one `ContactContact` entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the `RelatedABContactSearchCriteria` entity and the `CriteriaDef` elements that are defined for it in `search-config.xml` is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

SSN / TaxID field not editable if autocomplete triggered (CTC-2989)

Issue – The **SSN/Tax ID** field of a contact is not immediately editable after an address autocomplete populates an address field.

Workaround – After autocomplete fills the address field, the **SSN/Tax ID** field has a drop-down menu button to its right. Click the button to open the menu, and choose either **Delete** or **Enter New Value**.

Studio/Platform known issues for ContactManager 8.0.5

Long text in table cells can add white space to the right of the page (PL-28288)

Issue – In an editable list view, extremely long text entered in a single cell can cause additional white space on the right side of the page. Long text is text that occupies approximately the entire width of the screen. This issue occurs primarily in Chrome.

Workaround – If you expect users to enter large amounts of text into the cells of a column, configure the column to support text wrapping.

Administrative command-line tools cannot refresh WSDL (PL-29021)

Issue – Administrative command-line tools rely on web service implementation classes such as `MaintenanceToolsAPI.gs`. If you change the web service implementation classes, administrative tools might fail because the tool's WSDL does not match the server WSDL. Some changes do not affect the WSDL. For example, adding a `@WsiPermission` annotation.

Workaround – Do the following:

From the `ContactManager/bin` directory, at a command prompt type the following command:

```
gwab regen-soap-api
```

In Windows Explorer, copy the WSDL files from the location:

```
ContactManager/soap-api/wsi/wsd1
```

to:

```
ContactManager/admin/res/wsi
```

No entity changed event fired through a one-to-one entity relationship (PL-26224)

Issue – The existence of a one-to-one entity relationship between two entities prevents an entity change event from being fired, even if it is supposed to. For example, in ContactManager there is a foreign key from `ABContact` to `Address` for the primary address, and a one-to-one relationship from `Address` to `ABContact`. Because of the one-to-one link, no event changed event is fired for any change in the primary address.

More generally, if entity E1 has a one-to-one relationship to entity E2, no entity changed event is fired for E2 if a property on E1 changes.

Workaround – You might be able to take advantage of a secondary change caused by the initial change, with the secondary change triggering the event to be fired instead. In the previous ContactManager example, making sure that changes to all properties in `Address` cause a `History` record to be generated ensures that the entity changed event will be fired.

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

Alignment property does not work for iInputs in a Detail View (PL-29429)

Issue – Inputs in a detail view do not align to the right or center. They always align to the left, regardless of the value of the `align` property.

Workaround – If you need to right align a series of items such as a set of monetary values, put them into a list view instead of a detail view. You can then set the alignment as needed.

JMS inbound integration implementations must be OSGi or Java plugins (PL-32054)

Issue – Writing JMS inbound integration handlers requires casting an object to the type `javax.jms.Message`. In Gosu, for some app servers this fails with the message: **Caused by:**
`gw.lang.parser.exceptions.ErrantGosuClassException: GosuClass test.GosuJMSMessageReplyHandler has errors, and cannot be used at runtime.`

Workaround – Implement JMS handler code for inbound integration in Java. Guidewire recommends that for inbound integration Java code, implement your code as an OSGi plugin, not a standard (non-OSGi) Java plugin.

Guidewire ContactManager 8.0.4 release notes

These release notes contain the following topics:

- “Release Information for ContactManager 8.0.4” on page 359
- “Installing ContactManager 8.0.4” on page 359
- “Support” on page 21
- “Major issues and changes for ContactManager 8.0.4” on page 360
- “Improvements and resolved issues for ContactManager 8.0.4” on page 361
- “Known issues and limitations for ContactManager 8.0.4” on page 364

Release Information for ContactManager 8.0.4

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the Release Notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 8.0.4” on page 359.

This release of Guidewire ContactManager is 8.0.4.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 7.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 8.0.4

For general installation information, see “Installing ContactManager” on page 55.

For versions of ContactManager prior to 8.0.4 that you have skipped, see:

- “Guidewire ContactManager 8.0.3 release notes” on page 366
- “Guidewire ContactManager 8.0.2 release notes” on page 377
- “Guidewire ContactManager 8.0.1 release notes” on page 388
- “Guidewire ContactManager 8.0.0 release notes” on page 400
- “Guidewire ContactManager 7.0.7 release notes” on page 405
- “Guidewire ContactManager 7.0.6 release notes” on page 412
- “Guidewire ContactManager 7.0.5 release notes” on page 416
- “Guidewire ContactManager 7.0.4 release notes” on page 423
- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include sections for ContactCenter release notes. For example, see the release note archive in the *New and Changed Guide*.

Upgrade Information for ContactManager 8.0.4

Incorrect warning message when running Configuration Upgrade Tool (PL-28723)

When upgrading from one ContactManager 8.0.x release to another ContactManager 8.0.x release, running the Configuration Upgrade Tool may produce following warning message:

```
WARN cannot find Emerald base configuration zip, this could indicate a problem...
```

If the ContactManager/modules/base.zip file does exist, then you can safely ignore this warning.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 8.0.4

This topic contains major issues and changes that can affect your installation.

New and changed features for ContactManager 8.0.4

For information on new features and other changes, see the topic “New and changed features in ContactManager 8.0.4” on page 35.

Base PCF file changes for ContactManager 8.0.4

The following link requires that the ReleaseNotes_files directory be on your local disk in the same directory as this release notes file.

ContactManager release 8.0.3 to 8.0.4

To view a report of the changes to the base PCF files, refer to the original ContactManager 8.0.4 release notes.

Base rule changes for ContactManager 8.0.4

The following link requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

ContactManager release 8.0.3 to 8.0.4

To view a report of the changes to the base rule files, refer to the original ContactManager 8.0.4 release notes.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 8.0.4

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ContactManager 8.0.4 improvements and resolved issues

ID	Description
Core	
CTC-3007	There was an issue with approval of contact changes involving related contacts causing ContactManager to throw a null pointer exception. This issue has been fixed.
CTC-3101	The upgrade trigger for adding ABContactTag entities has been adjusted to no longer create ABContactTag entities for retired ABContact entities. A version check has been added to the upgrade trigger to ensure that there are no duplicate LinkID values in the ABContact table.
CTC-3173	An issue with a stack overflow error on the Pending Changes screen caused by a large number of related contacts has been fixed.
CTC-3222	The AuthenticationServicePlugin.gwp file has been added to the plugin registry folder.
CTC-3248	There was an issue with updates that were rejected by ContactManager if the update involved addresses that had been made primary and then secondary again. This issue has been fixed.

Platform improvements and resolved issues for ContactManager 8.0.4

ID	Description
Appserver Support - WebSphere	
PL-31805	EAR built for WebSphere now contains an IBM-specific descriptor that sets <code>classloading</code> mode to <code>parent-last</code> and <code>WAR classloader</code> policy to <code>single</code> . Setting these parameters in WebSphere manually was required for prior releases. Verify that <code>classloading</code> mode and <code>WAR classloader</code> policy are set correctly after the installation.
Configuration Upgrade	
PL-32082	The configuration upgrade tool would report fatal errors with case issues in display keys. The tool now corrects these issues and proceeds with the configuration upgrade.
Database Instrumentation	

ID	Description
PL-25165	Guidewire has modified the Database Table Info download to include the XML database configuration files. To access the files, download the table information file from the Server Tools → Database Table Info screen, then open the <code>index.html</code> file, and then click config_files: Directory with config files .
PL-31967	Added two new options to the <code>table_import</code> command to list and download load history reports.
Database Support - Oracle	
PL-31514	LOB objects associated with spatial columns are now stored in a LOB tablespace, if so configured for the table.
Database Support - SQL Server	
PL-31729	LOB objects associated with spatial columns are now stored in a LOB tablespace, if so configured for the table.
PL-32898 CTC-3266	Guidewire has improved the performance of the production of the (Server Tools Info Pages) SQL Server DMV Snapshot by restricting the data extract to the top 400 queries.
Database Upgrade	
PL-32115	A SQL Server upgrade that runs only the version trigger that converts integers to big integers for ID and foreign keys now runs without error.
PL-32605	Resolved an issue that caused a <code>ClassCastException</code> exception in document generation if you set <code>IDocumentTemplateSource.gwp</code> plugin parameter <code>cacheDescriptors</code> to <code>true</code> .
Gosu	
PL-30881	Resolved an issue with GosuProgram threads retaining type system locks unnecessarily, which resulted in deadlock.
PL-32490	Fixed an issue where Gosu does not automatically downcast if the left side of the <code>typeis</code> or <code>typeof</code> expression uses deprecated members.
IntelliJ IDE - Typelist Editor	
PL-30524	Fixed an issue where Studio would encode files containing non-English characters as ANSI.
Messaging	
PL-31680	Resolved a thread safety issue that occurred in rare cases when initializing event messages.
Other	
PLWEB-161	Fixed an issue where printing list views would throw an exception.
PLWEB-1997	Fixed an issue where an ampersand or other special character would break auto-filling other data.
PLWEB-2032	Added a width attribute to <code>ToolBarFilter</code> to specify a width in pixels. This allows you to increase the width of a <code>ToolBarFilter</code> if the default width does not allow all filter options to be displayed.
PLWEB-4568	Restored the ability to hide the sidebar.
PLWEB-5	Fixed an issue where the progress bar could be triggered multiple times and cause an incorrect status to be shown.
Other - Cloud	
PL-27758	CSV export from a list view now correctly handles newline characters (CR, LR) in the source data.
PL-32043	Upgraded the Joda-Time library to version 2.5 for updated time zone information.
Other - Integration Pod	
PL-31798	The utility class instance for the production data change utility methods changed packages. If you use the access syntax that the documentation recommends (<code>DataChange.util</code>), your code requires no changes.
PCF - Layout - List View	
PLWEB-2017	In a <code>RowTree</code> , a <code>Row</code> widget that has its <code>highlighted</code> property set to <code>true</code> now displays as highlighted.
Staging Tables	

ID	Description
PL-30911	(Oracle) Guidewire has added a <loader> element to database-config.xml that you can use to configure parallel operations during staging table load.
PL-31439	ClaimCenter and BillingCenter provide a configuration to run staging table based data loading with parallelism on Oracle.
PL-31931	If an integrity check and load fails after the check, another load operation on the same database is not performed.
Web - Configuration	
PLWEB-2000	Fixed an issue where the regional calendar for Portuguese displayed extraneous characters.
PLWEB-2035	Resolved an issue where the combination date-time widget would not auto-fill with a standard time when only the date was entered, if the date contained leading zeroes in the month or day.
PLWEB-4251	Added a visible attribute to the TabBarUnsavedWork and the MenuActions button to allow dynamic enabling and disabling of their visibility based on Gosu logic.
Web - IE Support	
PLWEB-18	As a security precaution, added an explicit header to prevent MIME sniffing.
Web - Styling	
PLWEB-1993	Fixed an issue where dividers were not appearing in the Actions menu.
PLWEB-4289	Fixed an issue where the labelStyleClass was not being applied properly to inputs in the GlobalAddressInputSet.
Web - UI/Runtime	
PLWEB-140	Fixed an alignment issue with check boxes in input groups.
PLWEB-15	Fixed an issue where dates with leading zeroes in the month and year would not autoformat correctly for all localization options.
PLWEB-1728	Improved keyboard navigation of the Actions menu.
PLWEB-1918	Fixed an issue where sorting a tree table would throw an exception.
PLWEB-2003	Fixed an issue where an unsecured GET request could be exposed.
PLWEB-2005	Fixed an issue where pressing Enter on a check box would both toggle the check box and submit the page.
PLWEB-2010	Fixed an issue where numCols did not appropriately scale for the font size. For example, year fields specified with numCols = 4 did not show all four digits. This may affect the layout of screens where numCols has previously been set.
PLWEB-2034	Fixed an issue with the target property of the ExitPoint widget that prevented it from functioning properly.
PLWEB-2039	Pressing Enter no longer changes the value of a check box. Now only pressing Space changes it.
PLWEB-2040	Fixed an issue where a button with the download attribute set to true lost the chosen values from the multi-select shuttle widget.
PLWEB-2047	Fixed an issue where help text would not appear for boolean radio button inputs.
PLWEB-2079	Fixed the info bar to allow for better copying and pasting of individual text snippets.
PLWEB-2191	Modified info bar styling to better reflect which items are links instead of text.
PLWEB-3042	Modified styling of input groups to better delineate expansion and separation.
PLWEB-32	Fixed an issue where inputs would not reset to the default value after enabling and disabling.
PLWEB-37	A date and time entered without separators, such as "11272014 1035am", is now accurately translated and validated. Note that a space must separate the date from the time.
PLWEB-4199	Fixed an issue where the objFocusID was not escaping HTML properly.

ID	Description
PLWEB-4521	Fixed an issue where the HTML in a RangeCell was not escaping properly.
Web Services - WSI (New)	
PL-32548	Fixed a security vulnerability to XML external entity (XXE) attacks. For more information about this issue, or to learn about temporary workarounds while you deploy this fix, visit the Guidewire Community and search for Knowledge article <i>"Are Guidewire PolicyCenter, BillingCenter, ClaimCenter, and ContactManager Vulnerable to XML External Entity (XXE) Attacks in Their Web Service Layers?"</i> .
XMLNode	
PL-31602	Resolved an XML parsing issue due to the compatibility-xsd.xml file omitting an entry for gw.xml.xsd.xsdtypes.

Known issues and limitations for ContactManager 8.0.4

This topic describes known issues with this release of Guidewire ContactManager.

Note: For maintenance releases, Guidewire often defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 8.0.4 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one ContactContact entity. ContactManager does not verify that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant ContactContact entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one ContactContact entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the RelatedABContactSearchCriteria entity and the CriteriaDef elements that are defined for it in search-config.xml is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

SSN / TaxID field not editable if autocomplete triggered (CTC-2989)

Issue – The **SSN/Tax ID** field of a contact is not immediately editable after an address autocomplete populates an address field.

Workaround – After autocomplete fills the address field, the **SSN/Tax ID** field has a drop-down menu button to its right. Click the button to open the menu, and choose either **Delete** or **Enter New Value**.

Studio/Platform known issues for ContactManager 8.0.4

Long text in table cells can add white space to the right of the page (PL-28288)

Issue – In an editable list view, extremely long text entered in a single cell can cause additional white space on the right side of the page. Long text is text that occupies approximately the entire width of the screen. This issue occurs primarily in the Google Chrome browser.

Workaround – If you expect users to enter large amounts of text into the cells of a column, configure the column to support text wrapping.

Administrative command-line tools cannot refresh WSDL (PL-29021)

Issue – Administrative command-line tools rely on web service implementation classes such as `MaintenanceToolsAPI.gs`. If you change the web service implementation classes, administrative tools might fail because the tool's WSDL does not match the server WSDL. Some changes do not affect the WSDL. For example, adding a `@WsiPermission` annotation.

Workaround – Do the following:

From the `ContactManager/bin` directory, at a command prompt type the command:

```
gwab regen-soap-api
```

In Windows Explorer, copy the WSDL files from the location:

```
ContactManager/soap-api/wsi/wsd1
```

to:

```
ContactManager/admin/res/wsi
```

No entity changed event fired through a one-to-one entity relationship (PL-26224)

Issue – The existence of a one-to-one entity relationship between two entities prevents an entity change event from being fired, even if it is supposed to be. For example, in ContactManager there is a foreign key from `ABContact` to `Address` for the primary address, and a one-to-one relationship from `Address` to `ABContact`. Because of the one-to-one link, no event changed event is fired for any change in the primary address.

More generally, if entity E1 has a one-to-one relationship to entity E2, no entity changed event is fired for E2 if a property on E1 changes.

Workaround – You might be able to take advantage of a secondary change caused by the initial change, with the secondary change triggering the event to be fired instead. In the previous ContactManager example, making sure that changes to all properties in `Address` cause a history record to be generated ensures that the entity changed event will be fired.

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

New mechanism for reloading Gosu classes (DOC-8218)

Issue – In past releases, you could modify your PCF files and Gosu classes in Guidewire Studio™, and then reload the changes into the running server by pressing `Alt+Shift+L` in the application user interface. This shortcut no longer loads Gosu classes.

Workaround – To have the server reload your Gosu classes, in Studio, click **Build→Make Project**. When Studio is finished compiling your project, the changes will be loaded. You can also restart your server to load the Gosu classes.

Alignment Property Does Not Work for Inputs in a DetailView (PL-29429)

Issue – Inputs in a detail view do not align to the right or center. They always align to the left, regardless of the value of the `align` property.

Workaround – If you need to right align a series of items such as a set of monetary values, put them into a list view instead of a detail view. You can then set the alignment as needed.

JMS inbound integration implementations must be OSGi or Java plugins (PL-32054)

Issue – Writing JMS inbound integration handlers requires casting an object to the type `javax.jms.Message`. In Gosu, for some app servers this fails with the message: “Caused by: gw.lang.parser.exceptions.ErrantGosuClassException: GosuClass test.GosuJMSMessageReplyHandler has errors, and cannot be used at runtime.”

Workaround – Implement JMS handler code for inbound integration in Java. Guidewire recommends that for inbound integration Java code, implement your code as an OSGi plugin, not a standard (non-OSGi) Java plugin.

Guidewire ContactManager 8.0.3 release notes

These release notes contain the following topics:

- “Release Information for ContactManager 8.0.3” on page 366
- “Installing ContactManager 8.0.3” on page 366
- “Support” on page 21
- “Major issues and changes for ContactManager 8.0.3” on page 367
- “Improvements and resolved issues for ContactManager 8.0.3” on page 368
- “Known issues and limitations for ContactManager 8.0.3” on page 375

Release Information for ContactManager 8.0.3

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the Release Notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 8.0.3” on page 366.

This release of Guidewire ContactManager is 8.0.3.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 7.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 8.0.3

For general installation information, see “Installing ContactManager” on page 55.

For versions of ContactManager prior to 8.0.3 that you have skipped, see:

- “Guidewire ContactManager 8.0.2 release notes” on page 377
- “Guidewire ContactManager 8.0.1 release notes” on page 388
- “Guidewire ContactManager 8.0.0 release notes” on page 400
- “Guidewire ContactManager 7.0.7 release notes” on page 405
- “Guidewire ContactManager 7.0.6 release notes” on page 412
- “Guidewire ContactManager 7.0.5 release notes” on page 416
- “Guidewire ContactManager 7.0.4 release notes” on page 423
- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include sections for ContactCenter release notes. For example, see the release note archive in the *New and Changed Guide*.

Upgrade Information for ContactManager 8.0.3

Incorrect warning message when running Configuration Upgrade Tool (PL-28723)

When upgrading from one ContactManager 8.0.x release to another ContactManager 8.0.x release, running the Configuration Upgrade Tool may produce following warning message:

```
WARN cannot find Emerald base configuration zip, this could indicate a problem...
```

If the ContactManager/modules/base.zip file does exist, then you can safely ignore this warning.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 8.0.3

This topic contains major issues and changes that can affect your installation.

New and changed features for ContactManager 8.0.3

For information on new features and other changes, see the topic “New and Changed Features in ContactManager 8.0.3” in the *Guidewire Contact Management Guide*.

Improvements to List View Navigation (PL-26518)

The following improvements have been made to the behavior of selecting and navigating within a list view:

- Selection now favors selecting a cell rather than a row. A row is still selected where required, such as when selecting a row in a list detail view.
- Use the arrow keys, **Tab** (next cell), and **Shift+Tab** (previous cell) to navigate within in a list view.
- With an editable cell selected, press **Enter** to begin editing it. To leave edit mode, press **Esc** or click outside of the list view.
- When editing a list view cell, press **Tab** or **Shift+Tab** to remain in edit mode and navigate to the next or previous editable cell.
- For cells that have special controls, use standard keyboard actions to change the cell value. For example, press **Space** or **Enter** to select a check box or radio button, press **Down Arrow** to activate a drop-down list, and so on.

Base PCF file changes for ContactManager 8.0.3

The following link requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

[ContactManager release 8.0.2 to 8.0.3](#)

To view a report of the changes to the base PCF files, refer to the original ContactManager 8.0.3 release notes.

Base rule changes for ContactManager 8.0.3

The following link requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

[ContactManager release 8.0.2 to 8.0.3](#)

There are no changes to the base rules.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 8.0.3

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ContactManager 8.0.3 improvements and resolved issues

ID	Description
Core	
CTC-1709	An error message was not accurately reflecting the required minimum search criteria. The message returned to the user now states that searching for a contact that has services requires specifying the requested services and either state or postal code. If the country being searched does not have any states, then only the services and postal code are required.
CTC-2794	There was an issue with pending updates in ContactManager—the edited elements were not being updated properly in the XML data. This issue has been fixed.

CTC-2824	An application can specify an external ID to be used by ContactManager as the LinkId value for a new contact. ContactManager already supported that functionality, and now it can also use an external ID to apply pending updates to entities created as a result of a different pending update.
CTC-2980	The Pending Changes screen in ContactManager now uses a cache for DiffDisplay objects. These cached objects contain the display information for the changes being made as part of a pending update from an external system. There are new preupdate rules for ABContact and PendingContactChanged to remove entries in the cache for an ABContact object that has changed. There are also two new configuration parameters: <ul style="list-style-type: none"> • MaxDiffDisplaysInCache – Controls the maximum size of the cache • DiffDisplaysCacheTimeoutInMinutes – Sets the amount of time since last access, in minutes, that a DiffDisplay object times out of the cache

Platform improvements and resolved issues for ContactManager 8.0.3

ID	Description
Authentication	
PL-30775	Removed the unused configuration parameter ShouldSynchUserRolesInLDAP.
Batch Processes	
PL-30718	The free-text batch load command now works correctly with JNDI data sources.
Build Infrastructure	
PL-27679	The <code>gwab regen-java-api</code> command now generates Java API Javadoc into the <code>java-api/doc/api</code> directory.
Command Line Tools	
PL-27608	Removed extraneous files from the output of the command <code>zip-changed-config</code> . The list used is in <code>com.guidewire.tools.upgrade2.merge.ConfigMergeList#EXCLUDE_DIRS</code> . Also added new option <code>-e</code> , which enables you to specify a semicolon-separated list of directories to exclude.
Consistency Checker	
PL-30098	Fixed an issue where Database Consistency Checks that checked date parameters would compare against an incorrect date.
Database Support - DB2	
PL-30469	Replaced parameter <code>identifyQueriesViaComments</code> in <code>config.xml</code> with new parameters <code>IdentifyQueryBuilderViaComments</code> and <code>IdentifyORMLayerViaComments</code> . The <code>IdentifyQueryBuilderViaComment</code> parameter instruments high-level database objects constructed by using the query builder APIs. <code>IdentifyQueryBuilderViaComment</code> is set to <code>true</code> in the base configuration. The <code>IdentifyORMLayerViaComments</code> parameter instruments lower level objects, such as beans, typelists, and other database building blocks. <code>IdentifyORMLayerViaComments</code> is set to <code>false</code> in the base configuration.
Database Support - Oracle	
PL-29153	The AWR report will now only report activity for the targeted schema (Guidewire database) on an Oracle server hosting multiple schemas.
PL-29868	The configuration of Oracle range and hash partitioned indexes has changed from the last release. If you have any indexes configured for Oracle partitioning, read the topics on partitioning under “Configuring the Database” in the <i>Installation Guide</i> and make changes accordingly.
PL-30770	Updated upgrade options in <code>database-config.xml</code> files to show default values. Removed an unneeded setting.
PL-31513	Fixed an issue where Oracle LOBs were mistakenly set with the <code>noLogging</code> attribute.
PL-31548	If the AWR report experiences an XML parsing error while reading from <code>DBMS_SQLTUNE.REPORT_SQL_MONITOR</code> , the read of the data will be skipped and an explanatory message printed in the Errors section of the report.
PL-31810	Fixed a problem with Guidewire AWR download when the database is Oracle 12.1.0.2 patchset or higher.
Database Support - SQL Server	

ID	Description
PL-30578	Two pages have been added to the SQL Server DMV performance report that aggregate active session statistics by user and by action.
Email	
PL-27557	Added an additional constructor in <code>EmailTemplateSearchCriteria</code> to add a callback interface and set a variable when it does not exist in the current evaluation context. This is to cover cases where the environment in which the templates are retrieved (filtered) and the environment in which they are evaluated are not the same.
Entities/Metadata	
PL-30357	Fixed an issue where the <code>gwab regen-java-api</code> command would throw an exception when processing a type-list that had more than 4,000 typecodes. Now 8,000 typecodes are supported.
External Entities	
PL-28992	Resolved an issue that caused the <code>gwab regen-java-api</code> command to sometimes fail if the flag <code>-Ddeprecated=true</code> was specified.
Geocoding/Proximity Search	
PL-28585	Driving directions are now in the user's preferred language instead of the language of the locale for the user's preferred regional formats. For example, if the user's preferred language is French and preferred regional formats are Japan, driving instructions are in French.
Gosu	
PL-29715	Fixed an issue that could cause runtime exceptions if you set the <code>config.xml</code> parameter <code>WarnOnImplicitCoercion</code> to the value <code>false</code> .
IntelliJ IDE - Compiler	
PL-29534	Compilation errors for a widget template are now shown against the containing PCF file rather than against the template.
PL-30918	Fixed an exception that would occur when a PCF file contained a widget template that contained errors.
IntelliJ IDE - Entity Editor	
PL-29124	Fixed an issue where comment elements such as <code><!-- Comment --></code> inside an entity or typelist definition file were removed when edited in Studio.
IntelliJ IDE - Localization	
PL-30209	You can now switch the language in Studio without needing to restart. To do so, navigate to File → Settings → Guidewire Studio , and then under Language Settings , click the language.
IntelliJ IDE - PCF Editor	
PL-30102	Removed a non-functional link in the pop-up help text when hovering over a PCF widget in Studio.
IntelliJ IDE - Plugins	
PL-29776	If you configure Guidewire Studio to work with IDEA IntelliJ Ultimate, now the OSGi Plugin Editor also uses Ultimate.
IntelliJ IDE - Product Model Editor	
PL-29996	Fixed an issue that prevented creating Gosu enhancements for product model coverages.
IntelliJ IDE - Webservices Editor	
PL-28462	Fixed an issue with the Web Services Collection Editor in Studio where it was unable to download imported schemas correctly.
PL-29920	Fixed an error that would occur in the Web Services editor in Studio when clicking Fetch Updates when the <code>suite-config.xml</code> file contained multiple values for the <code>env</code> attribute.
Messaging	

ID	Description
PL-27870	For messaging use, there are Gosu APIs that provide a <code>HashMap</code> that exists across multiple rule set executions. The APIs are methods on the result object of Gosu expression <code>messageContext.SessionMarker</code> . Older methods <code>addToTempMap</code> and <code>getFromGetMap</code> manipulate a <code>HashMap</code> that exists uniquely for each messaging destination. For a hash map that exists across all rules for all destinations, use the new methods <code>addToSessionMap</code> and <code>getFromSessionMap</code> .
PL-28769	Destination ID, message ID, sender ID, and requester user name of retried and skipped messages are now logged.
PL-28952	On Oracle with this release, <code>ContactManager</code> includes Oracle SQL hints in queries that select messages for a message destination. Oracle SQL hints can improve query performance. You can use the new parameter <code>UseOracleHintsOnMessageQueries</code> in <code>config.xml</code> to enable and disable the use of Oracle SQL hints at run time.
PL-29139	Several changes for inbound integration APIs: <ul style="list-style-type: none"> The <code>InboundIntegrationPlugin</code> plugin interface is now called <code>InboundIntegrationStartablePlugin</code> For messaging use, the new plugin interface <code>InboundIntegrationMessageReply</code> shares features of the inbound integration plugin but is a message reply plugin not a startable plugin Existing file and JMS integrations are now available also as <code>MessageReply</code> variants. See the <i>Integration Guide</i> topic “Multi-threaded Inbound Integration”.
PL-30119	Fixed an issue with the JMS inbound integration APIs that caused the exception “JMS Integration Service not supported on server runtime”.
PL-30689	The application no longer performs redundant locking and unlocking of entities if a message destination does not use the optional <code>MessageRequest</code> plugin.
Other - Developer IDE Pod	
PL-27489	Reduced the number of unnecessary library files that <code>regen-java-api</code> generates into the <code>lib</code> folder.
Other - Integration Pod	
PL-27845	Custom implementations of the inbound integration plugin interface (<code>InboundIntegrationPlugin</code>) now support Gosu implementations.
PL-30148	For use with the <code>DataChangeAPI</code> web service, new Gosu APIs for your data change code configure logging to the data change user interface. To log field-level entity changes, call <code>DataChange.util.setDetailResultWriting(bundle)</code> . To log arbitrary text, call <code>DataChange.util.ResultsWriter.append("your message")</code> .
PL-30566	The multi-threaded inbound integration <code>inbound-integration-config.xml</code> file now supports configurations that vary based on server environment set with the <code>env</code> system configuration setting. See the <i>Integration Guide</i> topic “Inbound Integration”.
PL-30910	For custom implementations of the <code>InboundIntegrationStartablePlugin</code> plugin interface, all method signatures of the start and stop methods must call methods of <code>gw.api.integration.inbound.CustomWorkAgent</code> . As a method argument, pass the plugin name that you used in your <code>inbound-integration-config.xml</code> file with the name attribute on the <code><custom-integration></code> element. In each start method, call <code>CustomWorkAgent.startCustomWorkAgent(pluginName)</code> . In each stop method, call <code>CustomWorkAgent.stopCustomWorkAgent(pluginName)</code> . This new API does not apply to writing handlers for the built-in file or JMS integrations.
Other - Persistence Pod	
PL-28950	This release improves the performance of the Populate Search Columns batch process.
PL-30644	Added configuration parameter <code>BatchProcessPopulateSearchColumnsParallelDML</code> to enable or disable parallelism in the populate search columns batch process. By default parallelism is enabled.
Other - Web UI Pod	
PL-29767	There was an issue with localized date formats that used a single-digit day or month, which sometimes resulted in swapping the day and month. Single digits are now supported for these values. In addition, the following date separators are supported by default: / (slash; example: 18/2/2015)

ID	Description
	. (dot; example: 18.2.2015) - (dash; example: 18-2-2015) <space> (example: 18 02 2015) <nospace> (example: 18022015)
PL-31050	Fixed an issue where the DateTime input widget was not navigable with the keyboard.
Queries	
PL-29695	Fixed the forEmpty method on Query objects to work properly on root queries and have no effect on subqueries.
Search	
PL-31446	Configuration of application servers changed due to changes in logging. See “Free-text Search Setup” in the <i>Installation Guide</i> for the modified setup procedures.
Web - Configuration	
PL-29869	Fixed an issue where, in the RowTree widget, a Cell with a currency format type or a CurrencyCell would not show the currency format properly.
PL-30310	Fixed an issue where validation on a PrivacyInput was applying to the mask value instead of the actual value.
PL-30475	Fixed an issue where the labelWidth property in an InputColumn was not being applied for specific widgets.
Web - IE Support	
PL-29681	Fixed an error where the server would report that the file toolbar-top-info-bg.gif was missing.
Web - ListViews	
PL-26386	Fixed an issue when postOnChange is enabled where tabbing through menu items or inputs with a menu icon would not show focus on those fields correctly.
PL-29294	Fixed an issue with a newly added list view column appearing at the extreme right until the layout preferences are reset.
PL-29657	Improved the behavior of tabbing through list view cells.
PL-29769	Fixed an issue where header menus were appearing incorrectly for list views with the colspan attribute set.
PL-30595	Fixed an issue where the screen would reset its horizontal scroll position while editing a list view in Internet Explorer.
PL-30989	Fixed an issue where the helper icon in a cell would not be visible when the cell had focus.
PL-31015	Fixed an issue where a LinkCell was not accessible with the keyboard.
PL-31038	List views now reposition themselves into view horizontally if you use the keyboard to navigate to an element off the screen.
PL-31152	Fixed an issue where you could not use the keyboard to invoke actions on text in list view cells.
PL-31201	Fixed an issue where the PrivacyInput menu remained expanded when navigating the page using the keyboard.
PL-31298	Fixed an issue where you could not open the menus of a ButtonCell with a MenuItem by using the Alt+Down Arrow keyboard shortcut.
PL-31354	Fixed an issue in list views where postOnChange would cause the focus to be set incorrectly.
PL-31380	Fixed an issue where focus would be lost if a postOnChange occurred in the last cell of a list view.
PL-31403	Fixed an issue where postOnChange was not triggered in list views with only one editable cell in Internet Explorer.
Web - Other	

ID	Description
PL-28929	Fixed an error message that referenced a missing “perf-analyzer”. The analyzer is an internal Guidewire tool that is not used in customer releases.
PL-30507	Fixed an issue in the RunBatchProcessCommand to display an error instead of a crash if a work queue contains zero workers.
PL-31341	Fixed an issue where the no-store value was omitted from the Cache-control HTTP response header in application responses.
PL-31813	Fixed an issue where calling an ExitPoint would fail.
Web - Styling	
PL-28575	The workspace panel had a property that enabled it to float while hovering. This behavior led to a confusing user experience because the panel would appear to pop up and down on user actions. The hover behavior has been disabled, and a larger button for expand/collapse has been added.
PL-30112	Fixed an issue where the PreFormattedTextInput PCF widget did not properly render HTML.
PL-30534	Fixed an issue where headings in the Actions menu were not appearing in a bold font.
Web - UI/Runtime	
PL-28186	Fixed an issue where the PrivacyInput field would be exposed if used during postOnChange.
PL-28224	Fixed an issue where the PrivacyInput widget was not masking values that were changed by client-side reflection.
PL-28581	Fixed an issue where actions in the TitleBar were not displayed properly.
PL-28784	Fixed an error that occurred in scatter charts after data axes were hidden.
PL-29118	To prevent CSRF (Cross-Site Request Forgery) attacks ContactManager now generates a unique Cross-Site Request Forgery (CSRF) token for each user session. The CSRF token is included in each request and used by the server to verify the legitimacy of the user request. See “Configuring Single Sign-on Authentication” in the <i>Installation Guide</i> .
PL-29230	Fixed a localization issue where the shadow placeholder text in the QuickJump box did not immediately refresh after the language was changed.
PL-29650	Fixed RangeValueInput to handle values that require HTML encoding (for example, an ampersand).
PL-29653	Fixed an issue where the screen would freeze or become blank when using typeahead in an AddressAutoFillInput widget.
PL-29689	Fixed an issue where detail views would lose focus after a postOnChange.
PL-29805	Fixed an issue where pressing Alt+Z to log out would do so even if there was unsaved work.
PL-29858, PL-30039	Fixed an issue with list views unexpectedly scrolling to the top after interacting with a radio button, check box, or combo box in a row.
PL-30055	Addressed an issue where keyboard input entered while a postOnChange event was in progress disappeared after the event completed. Keyboard input is now disabled until the event is complete.
PL-30072	Fixed an issue where postOnChange in the date widget required an additional user action before it would be triggered.
PL-30160	Fixed an issue where opening and then closing the DateTime widget would reset the minute value to the current minute of the user's machine.
PL-30207	Fixed an issue where you could not navigate out of a list view by using the Tab key.
PL-30492	Fixed an issue where using the Unsaved Work menu failed when a page title contained certain special characters (such as an ampersand).
PL-30586	Pressing the Backspace key no longer acts as a Back navigation command in the web browser.
PL-30772	Fixed an issue where the focus was not being set into a list view after a row was added.

ID	Description
PL-31001	Fixed an issue where the focus in a list view did not go to a newly added row.
PL-31027	Fixed an issue where list view rows with colspans specified did not handle tabbing correctly.
PL-31040	Fixed an issue where list view rows were highlighted incorrectly after columns were rearranged.
PL-31042	Fixed an issue where a list detail view configured to have an editable list view would create conflicting edits that would override each other and cause the data to be discarded. An impact of this change is that you now single-click a row to select it and double-click to edit it.
PL-31512	Fixed an issue where the calendar widget did not properly display Russian months.
PL-31610	Fixed an issue where reflection on the PCF widgets RangeRadioInput, BooleanRadioInput, and TypekeyRadioInput were not working properly.
Web Services - WSI (New)	
PL-31055	Fixed an issue in WSDL generation for services with multiple @Throws annotations for the same exception but different reasons. This issue caused an exception in some cases for WSDL consumers. The WSDL now merges related faults into one fault with all explanations concatenated in one comment. The fix prevents an exception that some WSDL consumers experienced.
Work Queues	
PL-31032	Improved performance of the work queue framework in finding available work items for work workers based on work item priority. The improvement introduces a new parameter, WorkItemPriorityMultiplierSecs, in config.xml, with a default value of 600 seconds.
PL-31043	Added BulkInsertWorkQueueBase for work queues with writers that typically select large volumes of work items. Instead of developing the work queue writer by implementing the findTargets method, implement the buildBulkInsertSelect method.
XMLElement (and XSD types)	
PL-28345	XSD types based on XmlNode have been deprecated, and will be removed in a future release.
PL-30793	Fixed an issue in WSDL import relating to handling simple data if a complexType is a complexContent extension of a complexType with a simpleContent extension.

Documentation improvements and resolved issues for ContactManager 8.0.3

ID	Description
DOC-2765	Added documentation on developing custom work queues. See “Custom Batch Processing” in the <i>Integration Guide</i> .
DOC-7022	Concepts and configuration of address autofill and autocompletion are now described in the topic “Address Autocompletion and Autofill” in the <i>Globalization Guide</i> .
DOC-7508	How to localize vendor services is now documented in the topic “Localizing Vendor Services” in the <i>Globalization Guide</i> .
DOC-8027	Added a topic that has steps for adding a new address field. This example covers the classes, configuration files, and PCF fields that are involved in adding a new address field. Additionally, the example shows how to configure autofill and autocomplete functionality for the new field. See “Example: Adding a Country with a New Address Field” in the <i>Globalization Guide</i> .
DOC-8646	Added documentation describing how to localize Contact name information to “Configuring Name Information” in the <i>Globalization Guide</i> .
DOC-8869	Added documentation for how to edit, configure, and load zone data. See “Configuring Zone Information” in the <i>Globalization Guide</i> .
DOC-8672	Enhanced documentation on debugging the application within Studio. See “Debugging and Testing Your Gosu Code” in the <i>Configuration Guide</i> .

DOC-8873 Added documentation on viewing entity property values during debugging. See “Debugging and Testing Your Gosu Code” in the *Configuration Guide*.

Known issues and limitations for ContactManager 8.0.3

This topic describes known issues with this release of Guidewire ContactManager.

Note: For maintenance releases, Guidewire often defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 8.0.3 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not verify that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one `ContactContact` entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the `RelatedABContactSearchCriteria` entity and the `CriteriaDef` elements that are defined for it in `search-config.xml` is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

SSN / TaxID field not editable if autocomplete triggered (CTC-2989)

Issue – The **SSN/Tax ID** field of a contact is not immediately editable after an address autocomplete populates an address field.

Workaround – After autocomplete fills the address field, the **SSN/Tax ID** field has a drop-down menu button to its right. Click the button to open the menu, and choose either **Delete** or **Enter New Value**.

Studio/Platform known issues for ContactManager 8.0.3

Renaming method or property throws ParseResultsException (PL-16633)

Issue – If you rename a property or a method or you change a method signature, and a workflow references that property or method in a Gosu field, ContactManager throws `ParseResultsException`. This is the intended behavior.

Workaround – Restart the workflow engine. To do so:

1. Log in to ContactManager by using an administrative account.
2. Access **Internal Tools**→**Reload**.
3. Click **Reload Workflow Engine**.

Long text in table cells can add white space to the right of the page (PL-28288)

Issue – In an editable list view, extremely long text entered in a single cell can cause additional white space on the right side of the page. Long text is text that occupies approximately the entire width of the screen. This issue occurs primarily in the Google Chrome browser.

Workaround – If you expect users to enter large amounts of text into the cells of a column, configure the column to support text wrapping.

Administrative command-line tools cannot refresh WSDL (PL-29021)

Issue – Administrative command-line tools rely on web service implementation classes such as `MaintenanceToolsAPI.gs`. If you change the web service implementation classes, administrative tools might fail because the tool's WSDL does not match the server WSDL. Some changes do not affect the WSDL. For example, adding a `@WsiPermission` annotation.

Workaround – Do the following:

From the `ContactManager/bin` directory, at a command prompt type the command:

```
gwab regen-soap-api
```

In Windows Explorer, copy the WSDL files from the location:

```
ContactManager/soap-api/wsi/wsd1
```

to:

```
ContactManager/admin/res/wsi
```

No entity changed event fired through a one-to-one entity relationship (PL-26224)

Issue – The existence of a one-to-one entity relationship between two entities prevents an entity change event from being fired, even if it is supposed to be. For example, in `ContactManager` there is a foreign key from `ABContact` to `Address` for the primary address, and a one-to-one relationship from `Address` to `ABContact`. Because of the one-to-one link, no event changed event is fired for any change in the primary address.

More generally, if entity `E1` has a one-to-one relationship to entity `E2`, no entity changed event is fired for `E2` if a property on `E1` changes.

Workaround – You might be able to take advantage of a secondary change caused by the initial change, with the secondary change triggering the event to be fired instead. In the previous `ContactManager` example, making sure that changes to all properties in `Address` cause a history record to be generated ensures that the entity changed event will be fired.

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

New mechanism for reloading Gosu classes (DOC-8218)

Issue – In past releases, you could modify your PCF files and Gosu classes in Guidewire Studio™, and then reload the changes into the running server by pressing `Alt+Shift+L` in the application user interface. This shortcut no longer loads Gosu classes.

Workaround – Have the server reload your Gosu classes. In Studio, click **Build→Make Project**. When Studio is finished compiling your project, the changes will be loaded. You can also restart your server to load the Gosu classes.

Alignment Property Does Not Work for Inputs in a DetailView (PL-29429)

Issue – Inputs in a detail view do not align to the right or center. They always align to the left, regardless of the value of the `align` property.

Workaround – If you need to right align a series of items such as a set of monetary values, put them into a list view instead of a detail view. You can then set the alignment as needed.

JMS inbound integration implementations must be OSGi or Java plugins (PL-32054)

Issue – Writing JMS inbound integration handlers requires casting an object to the type `javax.jms.Message`. In Gosu, for some app servers this fails with the message: “Caused by: gw.lang.parser.exceptions.ErrantGosuClassException: Gosu Class test.GosuJMSMessageReplyHandler has errors, and cannot be used at runtime.”

Workaround – Implement JMS handler code for inbound integration in Java. Guidewire recommends that for inbound integration Java code, implement your code as an OSGi plugin, not a standard (non-OSGi) Java plugin.

- Instruct users to manually supply a time along with the date.

DateTime widget does not auto-fill the time when the date contains leading zeroes (PL-32116)

Issue – The `DateTime` widget, a variation of the `DateInput` widget, configures both `dateFormat` and `timeFormat` for display. If you enter just a date into this field without entering a time, the application is supposed to auto-fill the time to 12:00 a.m. However, if the date specified contains leading zeros in both the month and day, such as 01/01/2014 for January 1, 2014, the time field is not auto-filled.

Workaround – A few workaround options are available for this issue:

- If users are not expected to enter the time, configure the widget to display only the date.
- Configure the display to show two separate date inputs, one for date and one for time.
- Inform users to not use leading zeros when entering the date. For example, 1/1/2014. In that case the time autofill works correctly.

Guidewire ContactManager 8.0.2 release notes

These release notes contain the following topics:

- “Release Information for ContactManager 8.0.2” on page 377
- “Installing ContactManager 8.0.2” on page 378
- “Support” on page 21
- “Major issues and changes for ContactManager 8.0.2” on page 379
- “Improvements and resolved issues for ContactManager 8.0.2” on page 379
- “Known issues and limitations for ContactManager 8.0.2” on page 385

Release Information for ContactManager 8.0.2

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the release notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 8.0.2” on page 378.

Version number

This release of Guidewire ContactManager is 8.0.2.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 7.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 8.0.2

For general installation information, see “Installing ContactManager” on page 55.

For versions of ContactManager prior to 8.0.2 that you have skipped, see:

- “Guidewire ContactManager 8.0.1 release notes” on page 388
- “Guidewire ContactManager 8.0.0 release notes” on page 400
- “Guidewire ContactManager 7.0.7 release notes” on page 405
- “Guidewire ContactManager 7.0.6 release notes” on page 412
- “Guidewire ContactManager 7.0.5 release notes” on page 416
- “Guidewire ContactManager 7.0.4 release notes” on page 423
- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include sections for ContactCenter release notes. For example, see the release note archive in the *New and Changed Guide*.

Upgrade Information for ContactManager 8.0.2

Incorrect warning message when running Configuration Upgrade Tool (PL-28723)

When upgrading from one ContactManager 8 release to another ContactManager 8 release, running the Configuration Upgrade Tool may produce following warning message:

```
WARN cannot find Emerald base configuration zip, this could indicate a problem...
```

If the ContactManager/modules/base.zip file does exist, then you can safely ignore this warning.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 8.0.2

This topic contains the following changes that might affect your installation. For information on new features and major changes, see the topic “New and Changed in ContactManager 8.0.2” in the Contact Management Guide.

- “Base PCF file changes for ContactManager 8.0.2” on page 379
- “Base rule changes for ContactManager 8.0.2” on page 379
- “Changes in this release provided in Upgrade Diff report” on page 311

Base PCF file changes for ContactManager 8.0.2

The link that follows requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

ContactManager release 8.0.1 to 8.0.2

To view a report of the changes to the base PCF files, refer to the original ContactManager 8.0.2 release notes.

Base rule changes for ContactManager 8.0.2

The link that follows requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

ContactManager release 8.0.1 to 8.0.2

There are no changes to the base rules.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 8.0.2

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ContactManager 8.0.2 improvements and resolved issues

ID	Description
Core	
CTC-1438	There are database consistency checks that ensure that ABPerson subtypes do not have a Name field set in addition to LastName and FirstName. This constraint is now being enforced on commit for ABPerson entities to ensure that the Name field is set to null before the entity is committed to the database.
CTC-2439	The exception TooLooseContactDuplicateMatchCriteriaException thrown from the Duplicate Contacts Finder batch process was incorrectly based on the type of Contact being checked. The result was that PersonDuplicateFinder could fail for an ABUserContact entity based on the criteria for ABPerson, but the error would indicate the criteria for ABUserContact. This exception has been fixed and now indicates the actual DuplicateFinder criteria being checked.
Database	
CTC-1801	The denormalized field for City is now being updated after a contact edit, and contact searches that use that field now work properly.

CTC-2510	There was an issue with retired contacts that were not being filtered from consistency checks related to shared addresses. Not filtering retired contacts sometimes resulted in erroneous consistency check failures after merging contacts. This issue has been fixed.
----------	---

Integration-CC

CTC-2443	A problem with the integration for proximity search between ClaimCenter 7.0.4p3 or higher and ContactManager 8 has been resolved.
CTC-2545	If ClaimCenter creates a contact in ContactManager, changes can be made to the contact in ContactManager as part of creation of the contact. Previously, those changes were not being transmitted back to ClaimCenter. Now, ClaimCenter synchronizes the contact in ClaimCenter after creation in ContactManager to ensure that any changes that are made in ContactManager are reflected in ClaimCenter.

Miscellaneous

CTC-2438	The upgrade from 8.0.1 to 8.0.2 now uses the class ABEmeraldToEmeraldConfigUpgraderStepList. This class was missing in previous versions and caused upgrade to skip the associated step.
----------	--

Platform improvements and resolved issues for ContactManager 8.0.2

ID	Description
Build Infrastructure	
PL-24111	When generating a WAR file using the build-war command and passing -Dconfig.war.dictionary=true, the data dictionary was not being generated. This issue is resolved.
Clustering	
PL-29135	<p>In the event of a serious network failure (such as a network switch failure) when a cluster is split into many parts, JGroups might have problems merging these parts into a single cluster after the network is restored. To solve this potential problem Guidewire implemented a JGroups watchdog timer that resets the JGroups channel if it thinks that the node is running outside of the cluster for a long time.</p> <p>The JGroups watchdog uses the following algorithm:</p> <ul style="list-style-type: none"> Coordinator of the cluster sends periodic heartbeat messages. Each message contains the list of the cluster members from the coordinator's point of view. The watchdog resets the channel if the node does not receive messages from its coordinator for a long time. Note that an incorrect heartbeat (one that does not include the node in the list of members) is equal to a missing heartbeat. <p>This feature can be configured by adjusting two new parameters in config.xml:</p> <ul style="list-style-type: none"> JGroupsWatchdogHeartbeatIntervalSecs - Coordinator node sends a heartbeat each interval. Other nodes reset JGroups channel if they do not receive the heartbeat from their coordinator for several intervals. The default value is 30 seconds. JGroupsWatchdogMissedHeartbeatsBeforeReset - The number of missed heartbeat messages after which a node resets its JGroups channel. The default is 10.
PL-29240	<p>Updated the default value for ClusterProtocolStack to a configuration that is more reliable in case of a major network failure.</p> <p>FD protocol was replaced with FD_ALL, MERGE2 was replaced with MERGE3.</p>
Configuration Upgrade	
PL-19432	The configuration upgrade tool now accepts Y or N keyboard input for dialog boxes.
PL-28639	The configuration upgrade tool now formats metadata extension files in a more consistent way.
Consistency Checker	
PL-26560	Added a progress bar to the consistency check page.
PL-26841	Fixed an issue where a check constraint error might not show the description of the check that failed.
PL-28361	If one or more consistency checks result in an SQL failure, then the run will be flagged to allow a rerun of just the checks that failed in a single thread.

PL-28579	Consistency checks are now executed by the worker threads individually instead of being grouped by table and consistency check type.
PL-28819	If an application server crashes while consistency checks are being run, when the server restarts, the worker threads restart, but some of the consistency checks will not be run.
Consistency Checker, Other - Database Pod	
PL-28612	Fixed an issue where a work queue could wait a long time if the server is restarted when an item is being processed.
Data Distribution	
PL-28620	Fixed an issue that would cause the data distribution process to fail.
Database Configuration, Database Instrumentation	
PL-28478	Database debug logging including SQL statements has been enabled for JNDI configurations.
Database Configuration, Database Support - General, Database Support - Oracle, Database Upgrade	
PL-28601	Resolved an issue in which the database statistics configuration in <code>config.xml</code> was not honored during upgrade. Statistics for tables that were specified to keep their statistics were being erroneously deleted.
Database Configuration	
PL-28660	The <code>validationQuery</code> attribute has been removed from the database configuration of the DBCP connection pool.
Database Configuration, Database Support - Oracle	
PL-28730	Updated the Database Parameters/Guidewire Database Config Statistics Settings page to show only the configured items.
Database Instrumentation	
PL-28236	The Server Tools Database Statistics page has been simplified. In addition, the code that compared the actual row counts with the calculated row counts and used that information to calculate the statistics has been eliminated. Statistics can be run unconditionally on all or a set of tables, or you can run <i>incremental</i> statistics, which uses the <code>incrementalupdatethresholdpercent</code> attribute of the <code>databasestatistics</code> configuration element, which defaults to 10 percent, to decide which statistics to run. Now the database statistics report reports all the update statistics statements that would be executed.
PL-28294	Fixed an issue where <code>system_tools</code> and <code>maintenance_tools</code> could be run from the command line without proper permissions.
Database Support - General	
PL-28655	Improved the performance of proximity searches.
Database Support - Oracle	
PL-29300	The <code>enable-all</code> setting of the <code>ora-parallel-dml</code> attribute on the upgrade element no longer forces parallel execution in Oracle for an <code>InsertSelectBuilder</code> that is used outside of upgrade. The <code>BeforeUpgradeInsertSelectBuilder</code> is still forced to use parallel execution if the <code>ora-parallel-dml</code> attribute is set to <code>enable-all</code> .
Database Support - SQL Server, Database Upgrade	
PL-28779	Fixed an issue that prevented upgrade to a ContactManager 8.0 maintenance release when the database type is SQL Server.
Database Support - SQL Server	
PL-28885	Fixed an issue that allowed users to create the ContactManager schema on a SQL Server system database.
Database Upgrade	
PL-28302	The Upgrade Info report is now generated even when the upgrade fails, if the upgrade is run in development mode. The report is generated in the default temporary-file directory specified by the system property <code>java.io.tmpdir</code> . On UNIX systems the default value of this property is typically <code>/tmp</code> or <code>/var/tmp</code> . On Microsoft Windows systems it is typically <code>c:\temp</code> .

PL-28447	If data model files were changed but the <code>extension.properties</code> version number was not updated, ContactManager would report an error when starting the server. This error did not include the file name of the extension file that included the data model change. The error has been updated to include the extension file name.
PL-28463	Resolved an issue that caused a <code>ClassCastException</code> when trying to delete a column by using the method <code>dropColumns</code> inside an <code>AfterUpgradeVersionTrigger</code> .
PL-28572	Improved the performance of the upgrader when typecodes have been removed.
PL-28613	Fixed the ordering of typelist IDs, which could create a race condition during schema verification while upgrading the database.
PL-28773	We now support database upgrade version trigger configuration options to cause select statements, particularly in <code>VersionChecks</code> , to use an Oracle hint to cause parallel execution in query (<code>SELECT</code>) statements.
PL-29200	If the upgrade configuration attribute <code>deferCreateArchiveIndexes</code> is not explicitly specified in the <code>database-config.xml</code> file, then its default value has been changed to <code>false</code> . See the Upgrade Guide topic “Deferring Creation of Archive Indexes” for information about this feature.
Entities/Metadata	
PL-29932	Added the following attributes to the <code>MonetaryAmount</code> entity: <code>createHistogram</code> , <code>createAmountHistogram</code> , <code>createCurrencyHistogram</code> .
Gosu	
PL-28349	Previous versions of Gosu reported a warning on ambiguous method calls. Ambiguous method calls can hide a logical bug in your code. Previously, the Gosu compiler selected the best matching method to remove ambiguity. This release changes behavior in two ways: <ul style="list-style-type: none"> Ambiguous calls are now an error instead of a warning. Studio now has a code inspection to identify and optionally fix any ambiguous code to previous Studio behavior. This inspection is disabled by default. To find and fix potential logical errors, Guidewire recommends that you run the inspection and carefully individually analyze every ambiguous call before applying any proposed fix.
PL-28663	You can now nest comments in Gosu code.
Gosu, IntelliJ IDE - Gosu Editor	
PL-28777	Fixed an issue where Studio would display invalid “implicit conversion to integer” warnings on PCF code that used Gosu reflection.
IntelliJ IDE - Customer Build	
PL-29861	The version number reported by Guidewire Studio now reflects the particular Studio version rather than the version of its companion Guidewire application. These numbers may be different.
IntelliJ IDE - Entity Editor	
PL-28783	Fixed an issue with changes to a delegate containing <code>MonetaryAmount</code> not being saved correctly.
PL-29211	Fixed an error that would occur when trying to open several typelist editors at the same time.
IntelliJ IDE - Entity Editor, IntelliJ IDE - Typelist Editor	
PL-28999	Removed the file name suffix option from the entity extension and typelist extension dialog boxes, since this is not a common operation, and removing the option simplifies the creation process. To restore this option, click File → Settings , then navigate to the Guidewire Studio → Metadata Editor page, and then set Show filename suffix on new extension dialog .
PL-29592	Fixed an error that would occur if you tried to create an entity or typelist extension when one already existed.
IntelliJ IDE - Formatting	
PL-29310	When viewing PCF files, entities, and typelists as XML in Studio, unicode characters now display according to their language native symbol, rather than as a unicode escape sequence.
IntelliJ IDE - Other	

PL-27965	The Compare to Base and Show Diff windows can now be resized and maximized.
PL-29172	Guidewire Studio is bundled with IntelliJ IDEA Community Edition. If you would like to run Studio with your own installation of IntelliJ IDEA Ultimate Edition, you can now do so. Only version 12.1.7 of IntelliJ IDEA Ultimate Edition is supported. To enable this, in the Guidewire product root directory, create a text file named <code>studio.ultimate</code> that contains the path to the IntelliJ IDEA Ultimate installation directory.
IntelliJ IDE - Typelist Editor	
PL-28289	Fixed a problem that occurred after installing a language pack. Typelists within the language pack were not being recognized as valid metadata files.
Internal/Server Tools Pages, Work Queues	
PL-28725	Implemented the Work Queue management bean. The Management Beans page now shows a management bean for each work queue running locally.
Localization	
PL-27731	The maximum length of a phone number extension can now be set in the Plugin Registry editor. The default length is 4. To set this value: <ol style="list-style-type: none"> 1. Open Guidewire Studio and navigate in the Project window to configuration→config→Plugins→Registry and double-click <code>IPhoneNormalizerPlugin.gwp</code>. 2. Add a new parameter by clicking the + button to the right of the Parameters pane. 3. For the Name, enter <code>extensionLength</code>. 4. For the Value, enter a number representing the length of the phone number extension.
PL-29815	There was an issue with out of memory exceptions with large data sets when running the phone normalizer upgrade batch process writer. This issue has been fixed. The writer now creates batch process work items in chunks, reducing total heap space required.
Other - Database	
PL-28631	Fixed an issue with <code>SystemToolsAPI.getUpdateStatsState</code> returning the wrong state.
Other - Developer IDE	
PL-27186	Studio now shows warnings for improper usages of internal types when the usage is in an <code>extends</code> or <code>implements</code> clause.
PL-28718	Improvements have been made in IntelliJ with OSGi Editor when you upgrade or move a Guidewire product to a new disk location. In IntelliJ with OSGi Editor, open your module settings. Click OSGi Bundle Facet . To the right of Guidewire product directory text field, click the Change button and set the new disk path. The tool updates IDE library dependencies and build properties.
Other - Integration	
PL-29002	This release adds a new servlet utility class <code>gw.servlet.ServletUtils</code> , which includes methods to get the signed in User object. Use any of three authentication types: <ul style="list-style-type: none"> • The session token from a logged-in user linking from a PCF page • HTTP Basic authentication headers • The name/password pair from custom headers You can use more than one API in your servlet. For example check the session token, and if it is not available use HTTP Basic authentication. See the Servlets chapter of the Gosu Reference Guide for details.
Other - Persistence	
PL-28762	Updated the exception for bean overlaps to include more information about the bean, including the type of the bean and the fixed ID and date range.
Plugins	
PL-28621	Failing to start a startable plugin during server start now prevents the server from advancing to the next run level.
Queries	
PL-27790	Resolved an issue that table aliases in subqueries were not generated with unique values.

PL-28194 With this release, the query builder APIs throw an exception whenever an effdated entity participates in a query through a join statement.

Revisioning

PL-27210 Fixed an incorrect error that was generated by the “Verifies foreign key reference to a subtype is to correct subtype” consistency check.

Templates

PL-27589 The WS-I web service TemplateToolsAPI changed to expand functionality to note and email templates, not just document templates:

- The four methods `validateAllTemplates`, `validateTemplate`, `validateTemplateInLocale`, and `listTemplate` have new names: `validateAllDocumentTemplates`, `validateDocumentTemplate`, `validateTemplateDocumentInLocale`, `listDocumentTemplate`. The old method names are deprecated.
- New methods for note templates: `validateAllNoteTemplates`, `validateAllNoteTemplatesInLocale`, `validateNoteTemplate`, `validateNoteTemplateInLocale`, `listNoteTemplate`
- New methods for email templates: `validateAllEmailTemplates`, `validateAllEmailTemplatesInLocale`, `validateEmailTemplate`, `validateEmailTemplateInLocale`, `listEmailTemplate`

Web - Configuration

PL-28965 In PCF files, when the `LinkCell` widget is used under the `RowTree` widget, the `id` attribute of the `LinkCell` widget is now required.

Web - ListViews

PL-25265 In the PCF widget `RowTree`, removed support for the `pageSize` attribute, since the widget is able to resize dynamically. The default value for this attribute is now 0, and any other value produces a validation error.

PL-26720 Added validation to `RowTree` to prevent using `Cell` types that cannot be rendered inside the `RowTree`. For example, `CheckBoxCell` cannot show check boxes inside the tree. If needed, convert existing usages to the type `Cell`.

PL-26900 Fixed an issue in `ListView` widgets. Hidden columns that were re-displayed were not becoming editable when triggered.

PL-28169 Fixed an issue with user-hidden columns, which caused layout problems by disabling the ability to show/hide columns for columns when a `colspan` is specified.

PL-28782 Fixed an issue with the `RangeValueWidget`. `RowTree` cells were not displaying the proper localized value.

PL-29483 Fixed an issue with grouping a `ListView`, which displayed a generic `Object` string instead of the correct value.

Web - UI/Runtime

PL-27761 Fixed an issue with pressing `Enter` not properly triggering a search or navigation.

PL-28593 Fixed an issue with text in the `InfoBar` that was not selectable for copying.

PL-28666 Added validation to the `DetailViewPanel` configuration to ensure that `Input` widgets are enclosed by an `InputColumn`.

PL-28770 Resolved an issue with `MonetaryAmountCell` that prevented it from handling some currency formats correctly.

PL-28854 Fixed an issue with rendering of `LinkCells` in the `RowTree`.

PL-28860 You can now localize watermarks in the user interface for date and time fields.

PL-28946 Fixed the `DateInput` widget for the time to reflect the user's settings for 12-hour versus 24-hour format.

PL-29052 Resolved a critical security vulnerability to non-persistent cross-site scripting attacks.

PL-29475 You can no longer select **Group By This Field** in a list view when it is in edit mode.

Web Services - WSI (New)

PL-27467 Fixed an issue with `WsiAuthenticationException`, which was being declared in generated WSDLs, but was not actually being returned.

PL-28179 Fixed an issue with `WebServiceServlet`, which was waiting for a return to multmode once the runlevel was dropped to maintenance mode before responding to requests.

PL-28267 Web services now throw HTTP 500 for SOAP faults instead of HTTP 200.

Known issues and limitations for ContactManager 8.0.2

This topic describes known issues with this release of Guidewire ContactManager.

Note: For maintenance releases, Guidewire often defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 8.0.2 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not verify that the relationship does not already exist, and therefore can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one `ContactContact` entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the `RelatedABContactSearchCriteria` entity and the `CriteriaDef` elements that are defined for it in `search-config.xml` is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

Platform/Studio known issues for ContactManager 8.0.2

Renaming method or property throws `ParseException` (PL-16633)

Issue – If you rename a property or a method or you change a method signature, and a workflow references that property or method in a Gosu field, ContactManager throws `ParseException`. This is the intended behavior.

Workaround – Restart the workflow engine. To do so:

1. Log in to ContactManager by using an administrative account.
2. Access **Internal Tools**→**Reload**.
3. Click **Reload Workflow Engine**.

Javadoc command does not generate index file in expected location (PL-27679)

Issue – When you run `gwXX regen-java-api`, an `index.html` file is not created in `ContactManager/java-api/doc`.

Workaround – The command now generates Javadoc JAR files in `ContactManager/java-api/doc`. To view the Javadoc, add the JAR files to the Guidewire Studio™ project.

Long text in table cells can add white space to the right of the page (PL-28288)

Issue – In an editable list view, extremely long text entered in a single cell can cause additional white space on the right side of the page. Long text is text that occupies approximately the entire width of the screen. This issue occurs primarily in the Google Chrome browser.

Workaround – If you expect users to enter large amounts of text into the cells of a column, configure the column to support text wrapping.

Build tool regen-java-api can fail with -Ddeprecated=true (PL-28992)

Issue – If you run the `gwab regen-java-api` command with the flag `-Ddeprecated=true`, the command can fail.

Workaround – Specify `dev-deploy` as the value of the `depends` attribute, as follows:

1. Open the `build.xml` file in `ContactManager/modules/ant` in an editor.
2. Find the entry that starts as follows:

```
<target name="regen-java-api" depends="init"
```

3. Change the value of the `depends` attribute to `dev-deploy`, as follows:

```
<target name="regen-java-api" depends="dev-deploy"
```

4. Save the file and then run the `gwab regen-java-api` command.

Administrative command-line tools cannot refresh WSDL (PL-29021)

Issue – Some administrative command-line tools rely on web service implementation classes, such as `MaintenanceToolsAPI.gs`. Source files for these classes use the `@Export` annotation, which enables you to edit the file. In this release, the administrative command-line tools cannot refresh the WSDL for these classes. Any change to the web service implementation class that changes the WSDL can prevent the administrative tools from working. Therefore, the only changes you can make to these classes are changes that do not affect the WSDL. For example, you can add `@WsPermission` annotations to change the permissions without changing the WSDL.

Workaround – Guidewire is aware of this issue.

No entity changed event fired through a one-to-one entity relationship (PL-26224)

Issue – The existence of a one-to-one entity relationship between two entities prevents an entity change event from being fired, even if it is supposed to be. For example, in `ContactManager` there is a foreign key from `ABContact` to `Address` for the primary address, and a one-to-one relationship from `Address` to `ABContact`. Because of the one-to-one link, no event changed event is fired for any change in the primary address.

More generally, if entity `E1` has a one-to-one relationship to entity `E2`, no entity changed event is fired for `E2` if a property on `E1` changes.

Workaround – You might be able to take advantage of a secondary change caused by the initial change, with the secondary change triggering the event to be fired instead. In the previous `ContactManager` example, making sure that changes to all properties in `Address` cause a history record to be generated ensures that the entity changed event will be fired.

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

New mechanism for reloading Gosu classes (DOC-8218)

Issue – In past releases, you could modify your PCF files and Gosu classes in Guidewire Studio™, and then reload the changes into the running server by pressing **Alt+Shift+L** in the application user interface. This shortcut no longer loads Gosu classes.

Workaround – Have the server reload your Gosu classes. In Studio, click **Build→Make Project**. When Studio is finished compiling your project, the changes will be loaded. You can also restart your server to load the Gosu classes.

Database Consistency Checks - The date parameter to check against is not correct (PL-30098)

Issue – Database consistency checks that check date parameters compare against an incorrect date.

Steps to reproduce the problem:

1. Start ContactManager 8.0.2 on a fresh database.
2. Run database consistency checks. These checks report zero failures.
3. Create records for entities that store the date/time that they were created.
4. From server tools, run consistency checks again. Consistency check failures related to the timestamp are reported.

The creation date of an object cannot be after the time. The consistency checks fail because the creation time is compared to the time of the first consistency check run of the application server session instead of to the current database time.

Workaround – Ignore database consistency check errors caused due to timestamp. Or, restart the application server and run consistency checks for the results to be accurate.

Some PCF widgets in ContactManager are miscategorized in Studio (PL-23454)

Issue – Some PCF widgets in ContactManager are miscategorized in Guidewire Studio™. This means that on certain screens (such as wizard screens), widgets that you may want to use are not available in the toolbox.

Workaround – Find a usage of the missing widget on a different screen, and then copy and paste it to the screen that you are editing.

Validation for WS-I Web Services not invoked by Studio (PL-29136)

Issue – Guidewire Studio™ does not validate WS-I web service implementation classes.

Workaround – Before deploying new web service code, run the `gwab verify-resources` tool from the command line.

Custom inbound integration implementations require OSGi (PL-28956)

Issue – Custom implementations of the `InboundIntegrationStartablePlugin` or `InboundIntegrationMessageReply` plugin interface must use OSGi.

Workaround – Implement the plugin interface as an OSGi plugin.

JMS inbound integration connectivity problem (PL-30199)

Issue – The JMS inbound integration APIs throw exception “JMS Integration Service not supported on server runtime”.

Workaround – Guidewire is aware of this issue.

Screen elements might disappear when using the AddressAutoFillInput widget (PL-29653)

Issue – Quick use of the `AddressAutoFillInput` widget might lead to a JavaScript error, which causes screen elements to disappear.

Workaround – Refresh the page, and the elements will reappear.

Data might disappear while typing while a PostOnChange is firing (PL-30055)

Issue – While a PostOnChange is occurring, the screen might continue to appear responsive, but action or keystrokes captured during that time will be erased when the post returns. This issue is most noticeable on slow networks or with a large amount of processing occurring during the post.

Workaround – Evaluate your users' network speeds to determine how likely it is that this issue will impact them. This is most likely to affect screens with quick, expert data entry. To improve the speed of the post, set a narrow target for postOnChange, or set it to DATA_ONLY if the user interface does not need to change.

Minute selection in time picker resets to current time (PL-30160)

Issue – When you open a date/time picker, the minutes are reset to the current time's minutes. For example, if the current computer time is 12:37 and the currently set time in the picker is 5:50, then opening and closing the picker changes the time to 5:37 and marks it as an edit.

Workaround – Guidewire is aware of this issue.

Guidewire ContactManager 8.0.1 release notes

These release notes contain the following topics:

- “Release Information for ContactManager 8.0.1” on page 388
- “Installing ContactManager 8.0.1” on page 388
- “Support” on page 21
- “Issues and major changes for ContactManager 8.0.1” on page 389
- “Improvements and general issues for ContactManager 8.0.1” on page 390
- “Known issues and limitations for ContactManager 8.0.1” on page 397

Release Information for ContactManager 8.0.1

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you skipped one or more upgrade releases to ContactManager, be sure to read the release notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 8.0.1” on page 388.

This release of Guidewire ContactManager is 8.0.1.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 7.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 8.0.1

For general installation information, see “Installing ContactManager” on page 55.

For versions of ContactManager prior to 8.0.1 that you have skipped, see:

- “Guidewire ContactManager 8.0.0 release notes” on page 400
- “Guidewire ContactManager 7.0.7 release notes” on page 405
- “Guidewire ContactManager 7.0.6 release notes” on page 412
- “Guidewire ContactManager 7.0.5 release notes” on page 416
- “Guidewire ContactManager 7.0.4 release notes” on page 423
- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include sections for ContactCenter release notes. For example, see the release note archive in the *New and Changed Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Issues and major changes for ContactManager 8.0.1

This topic contains the following changes that might affect your installation. For information on new features and major changes, see the topic “New and Changed in ClaimCenter 8.0.1” in the ClaimCenter New and Changed Guide.

- “Base PCF file changes for ContactManager 8.0.1” on page 389
- “Rule changes for ContactManager 8.0.1” on page 389
- “Changes in this release provided in Upgrade Diff report” on page 311

Base PCF file changes for ContactManager 8.0.1

The link that follows requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

ContactManager release 8.0.0 to 8.0.1

To view a report of the changes to the base PCF files, refer to the original ContactManager 8.0.1 release notes.

Rule changes for ContactManager 8.0.1

The link that follows requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

ContactManager release 8.0.0 to 8.0.1

To view a report of the changes to the base rules, refer to the original ContactManager 8.0.1 release notes.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and general issues for ContactManager 8.0.1

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ID	Description
Application Server	
PL-27203	JBoss 6 deployment now works properly.
Archiving	
PL-26930	Modified the <code>restoreSeveredTransactionOffsetOnsetOnsetLinks</code> method so it can be called from Gosu.
Bundles and Transactions	
PL-22240	The touch method has been re-implemented. Please refer to the <i>Gosudoc</i> for detailed usage.
Cognos Integration	
PL-21153	Thread usage on the LDAP server is now managed more efficiently.
Command Line Tools	
PL-27378	Removed <code>debug-start</code> from the command-line options.
PL-28544	The obsolete command line command <code>copy-theme</code> has been removed.
Configuration Upgrade	
PL-27207	The log file of the upgrade tool now reflects the difference between platform upgrade triggers and application-specific upgrade triggers.
PL-27208	You can now expand all and collapse all in the upgrade tool.
PL-27209	You can now right-click to export the directory tree within the upgrade tool.
PL-27338	Fixed an issue in which running upgrade multiple times created nested rules folders.
PL-27744	Fixed an issue that occurred if you added any <code>ClaimContactInput</code> widgets that use <code>postOnChange</code> , or if you modified a base <code>ClaimContactInput</code> widget by adding <code>postOnChange</code> . In these cases, the <code>postOnChange</code> would not get upgraded to the new syntax.
Core	
CTC-473	In the Merge Contacts screen, the Ignore buttons for each row representing a duplicate contact pair have been replaced with check boxes. There is now an Ignore button in the toolbar for this screen that applies to all checked rows. If you select a check box for any row, clicking Ignore applies to that checked row. When you click Ignore , before the action takes place, ContactManager displays a confirmation message that enables you to cancel the action.
CTC-2361	The Find Duplicates process now varies based on the locale. For example, for the Japanese locale, there is no TaxID involved in the query for the Find Duplicates process, and there is no possibility for an exact match, only a potential match. This behavior matches the Find Duplicates behavior in ContactManager 7.0.
CTC-2399	When the user has not assigned a primary phone number for a contact, ContactManager assigns the first phone field that has a value to the primary phone field. ContactManager now uses the order work phone, then home phone, then cell phone to determine which field to assign. If no phone number is specified in any of these fields, ContactManager does not assign a primary phone number.
Data Distribution	
PL-25702	The Data Distribution page enables Download comparison Zip file and Download Combined Zip file options only if two executions are selected. In prior versions, the options were erroneously enabled even if there were not two executions selected.

Database Configuration

PL-28656 Fixed a problem that prevented the DBAuthenticationPlugin from working.

Database Support

PL-24417 The performance requirement to set the action="delete" attribute for ab_message in database-config.xml has been removed.

PL-27438 Added support for Oracle Date interval partitioning.

PL-28323 Fixed a bug which prevented scheduling of the Database Statistics process.

Database Support – Oracle

PL-26203 Nullable monetaryamount columns will use the Oracle default value feature during upgrade. This is a performance optimization.

PL-28085 The Oracle fast add column feature is used when adding monetaryamount amount columns to an entity or a subtype.

PL-28327 Guidewire added an option to switch off the Oracle adaptive optimization feature for Guidewire applications. See “Configuring Oracle Adaptive Optimization for ContactManager” in the Installation Guide.

Database Upgrade

PL-27016 Guidewire has added support for using Oracle's parallel DDL execution feature during upgrade. The createIndexInParallel attribute of the <upgrade> element has been replaced with the new degree-parallel-ddl attribute. See “Configuring Parallel DML and DDL Statement Execution” in the *Upgrade Guide*.

PL-27865 Guidewire resolved a rare issue in which the server would be able to start after an incomplete upgrade but subsequent upgrade attempts would fail.

PL-27918 The Database Upgrader now honors the database statistics configuration.

PL-28041 The updatestatistics attribute in database-config.xml now controls both deletion and collection of database statistics during upgrade.

PL-28122 The DeferredUpgradeTasks process is now profiled and appears under Guidewire Profiler. This process is used when deferring creation of archive indexes until after the upgrade. See “Deferring Creation of Archive Indexes” in the *Upgrade Guide*.

PL-28449 Fixed a bug that prevented handling of statistics on the ID column properly.

Document Management

PL-27480 Document templates no longer have size constraints.

PL-27577 You can now call the method DocumentsUtil.createNewDocument without needing the current user to have the doccreate permission. Use this method if you implement the IDocumentMetadataSource plugin.

Email

PL-27921 Due to potential cross-site scripting vulnerabilities, HTML is disabled as the content of an email. If you would like to continue to have HTML email and accept the risk of this vulnerability, you may remove the escaping of the subject and body in the document template.

Entities/Metadata

PL-19023 The *Data Dictionary* has been modified to show references for subtypes

PL-24743, PL-27611 *Data Dictionary* descriptions for core locale fields have been updated.

PL-25622 Fixed an issue in which MonetaryAmount appeared as two separate fields under Actual Amount in the *Data Dictionary* Data Entity View.

PL-25809 Fixed an issue in which typelist codes did not include documentation for the typecodes (name and description).

PL-27465 Added upgrade trigger to insert xmlns if missing on .eti files.

PL-27501 Added the ability to override and add keyfilters and typefilters by using extensions.xml.

PL-27819	Added a new <code>overlapTable</code> attribute to the <code>edgeForeignKey</code> and <code>localization</code> entities, which specifies that the entity implements the <code>OverlapTable</code> delegate.
Globalization	
PL-26606	Updated <code>GroupUserSearchDV.pcf</code> to support Japanese kanji fields in the standard way represented elsewhere in the application.
PL-27394	Values substituted in display keys that are of type <code>BigDecimal</code> , <code>Date</code> , and <code>IMoney</code> and its implementing classes, such as <code>MonetaryAmount</code> , are now properly formatted according to the regional formats in effect for users. For example, the substitution value "123456" is formatted as "123,456" for U.S. (English) or "123 456" for France (French).
PL-27609	Fixed an issue in which the display value for locale shifts when an admin user is changing his/her own language/locale when viewing the profile of another user.
PL-28068	The configuration parameter for overriding the default maximum width for labels moved from display key <code>ExtJS.Form.LabelWidth</code> to XML element <code>LabelWidth</code> , with attribute <code>width</code> specified in pixels. Use the <code>LabelWidth</code> element in the <code>language.xml</code> file for the language that you want to configure. For example, <pre><GWLlanguage code="de_DE" name="German (Germany)" typecode="de_DE"> <ExtJSSettings> <LabelWidth size="190" /> </ExtJSSettings> </GWLlanguage></pre>
Gosu	
PL-18217	A Gosu class now preserves annotations inherited from Java classes in the class hierarchy. Note that this change effectively restricts access by a subclass to features tagged with the <code>@InternalAPI</code> annotation.
PL-25700	Fixed an issue that caused errors on startup under very specific conditions related to the compilation of particular Gosu classes.
PL-27099	Gosu supports annotations on parameters in methods, properties, and constructors.
PL-27428	The Gosu language now provides limited support for the Java annotation <code>@SuppressWarnings</code> , which tells the compiler to suppress warnings. Use this annotation on declarations of a type, function, property, constructor, field, or parameter. Note that local variables do not support this annotation. You must pass a <code>String</code> value as an argument to indicate which warnings to suppress. Pass the argument "all" to suppress all warnings. Pass the argument "deprecation" to suppress deprecation warnings. For example, to suppress deprecation warnings in a Gosu class, add the annotation <code>@SuppressWarnings("deprecation")</code> on the line before the class declaration.
PL-27651	Gosu now recognizes the annotation <code>@java.lang.Deprecated</code> as a form of deprecation, in addition to <code>@gw.lang.Deprecated</code> and the <code>@deprecated</code> Javadoc tag.
Integration	
PL-28196	This release changed how to configure inbound multi-threaded integrations such as the built-in file and JMS integrations. In previous releases, you added configuration parameters in the Plugins registry in Studio. In this release, you set a single parameter <code>integrationservice</code> and then do the rest of the configuration in the new file <code>inbound-integration-config.xml</code> . Also, the API details for file and JMS integrations changed. There is a new plugin interface called <code>InboundIntegrationHandlerPlugin</code> . Also, the file integration now supports processing one file at a time, rather than one line at a time. For details, refer to the <i>Integration Guide</i> .
Integration - CC	
CTC-1918	To extend the related contact Integration in <code>ContactManager</code> , you must modify the classes <code>ABContactAPIRelatedContact</code> and <code>RelatedContactInfoContainer</code> . These classes are now marked <code>@Export</code> and can be modified.

CTC-2028	An intermittent problem was occurring with linking a contact from ClaimCenter. Sometimes the contact was left out of sync after a link request was sent to ContactManager. This problem has been fixed. ClaimCenter now always synchronizes after linking.
CTC-2112	Previously, when you initially added a contact to ClaimCenter that was stored in ContactManager, all properties on the contact brought over from ContactManager were saved in the ClaimCenter database. Even properties that were marked non-persistent were saved in ClaimCenter. This problem has been fixed, and now ClaimCenter does not save non-persistent properties. This fix applies to fields in the ClaimCenter class ContactMapper to which you have added <code>.withAffectsSync(false).withPersist(false).withMappingDirection(TO_BEAN)</code> .
CTC-2286	Previously, the first name was required when creating a contact of type Person or PersonVendor or a subtype of one of these contact types. With this change, the first name is no longer required for creating any contact. You can edit <code>ValidateABContactCreationPluginImpl</code> to change this and other contact creation requirements.
CTC-2323	There was a problem with the Duplicate Finder batch process throwing an exception if there was a contact in Pending Create status. This issue has been fixed.
IntelliJ IDE – Compiler	
PL-28346	Fixed a compilation error when compiling an entity or its extension if the entity had a subtype.
IntelliJ IDE – Debugger	
PL-27875	When debugging Gosu code, you can now browse the structure of an Entity object and inspect the property values stored within it. To enable this, in Guidewire Studio, click File→Settings , and then in the Guidewire Studio panel set Enhance Entities Visualization .
IntelliJ IDE – Display Key Editor	
PL-27971	Improved typing performance in the Display Key editor.
PL-28677	Fixed a bug in Studio where converting a string into a display key caused exceptions. The behavior has changed slightly, so now the locale folder is required to have a <code>display.properties</code> file already in it before it appears in the Create Display Key dialog or the Step Name Localizations tab in the Workflow editor.
IntelliJ IDE – Entity Editor	
PL-26336	Column validation has been re-enabled in Studio.
PL-26364	Fixed an exception in the Entity editor that occurred when attempting to override a read-only attribute.
PL-26375	Fixed an issue in the New Entity dialog in which the <code>viewEntity</code> was listing suggestions that were not applicable.
PL-26540	Added additional error notes to the Entity editor to highlight the parent elements if a child is invalid.
PL-27220	In the Entity editor in Studio, when editing the <code><tag></code> subelement of the <code><column></code> element, there is now a drop-down list showing available values.
PL-27454	Fixed an issue in which creating multiple entity extensions with suffixes would produce an exception.
PL-27715	Fixed an error that would occur when the <code>effDatedBranchType</code> attribute was not correct in an entity of type <code>effdated</code> .
PL-28089	In the Entity editor in Studio, you are now required to specify a value for the <code>nullok</code> attribute.
PL-28529	Fixed an error in Studio when creating an entity extension if there is a Java class under <code>src</code> .
IntelliJ IDE – Gosu Editor	
PL-19418	Fixed a compilation error during bytecode generation on certain annotations from Java source types.
PL-26640	Fixed an exception that would occur when entering <code>display</code> in a Gosu class.
PL-26866	The Gosu <code>using</code> clause syntax now has an additional feature for adding additional cleanup code. You can optionally add a <code>finally</code> clause that runs after the statement body, even if exceptions occur in the body of the <code>using</code> clause. See the <i>Gosu Reference Guide</i> for details.
PL-27135	Studio now shows additional warnings for improper usages of internal <code>gw</code> classes.
PL-27320	Fixed an issue in which pressing <code>Ctrl+O</code> threw an exception in Gosu.

PL-27724	Fixed an issue with some deprecated methods not being shown in strikethrough text in Studio.
PL-27873	Fixed an issue in which pasting code into Studio caused multiline statements to be concatenated and merged with comments.
PL-27893	Fixed an exception that was thrown when creating a new Gosu template.
PL-27943	Fixed an issue with some deprecated methods not being shown in strikethrough text in Studio.
PL-27944	The Gosu language has two new compound assignment operators, which are operators that apply an operation to a variable then re-assign the variable to the result. The new operator <code>&&=</code> performs the logical AND operation to the previous value. The new operator <code> =</code> performs the logical OR to the previous value. Both operators work with the primitive type <code>boolean</code> or the object type <code>Boolean</code> on either side of the operator. For example, suppose you have two <code>boolean</code> variables called <code>needsUpdate</code> and <code>flagTest</code> . The statement <code>needsUpdate = flagTest</code> has the meaning of <code>needsUpdate = (needsUpdate OR flagTest)</code> . Do not confuse these new operators with the other operators <code>&=</code> and <code> =</code> , which apply bitwise AND and bitwise OR operations.
PL-28019	Fixed a false compile error in the Gosu editor that manifested when a property getter or setter overrode a getter or setter in a superclass implemented in Java.
PL-28027	Gosu does not support numeric expressions in the <code>for</code> statement after the <code>in</code> keyword. The code: <pre>for (x in 10) {...}</pre> is illegal and must be upgraded with an interval such as: <pre>for (x in 0..10) {...}</pre> using the provided inspection in Studio.
IntelliJ IDE – Line of Business Editor	
PL-26590	The LOB tab has been removed from <code>.tti</code> files in Studio to prevent missing loss types.
PL-26595	Added the ability to more easily select multiple typekeys in Studio by using the <code>Ctrl</code> or <code>Shift</code> keys.
PL-26620	Retired typecodes are now shown in strikethrough text.
PL-26826	Fixed an issue in which Studio would throw an exception when a <code>categorylist</code> was added to a <code>LossType</code> typecode.
PL-26971	Fixed an issue in Studio in which removing a typecode from its parent also incorrectly removed it from all its other parents.
IntelliJ IDE – Other	
PL-27198	Fixed an issue in which Run commands in the QuickJump box did not work when the server was started from Studio.
PL-27862	Fixed an issue where Studio would not suggest types defined on XSD files when trying to create an enhancement.
IntelliJ IDE – PCF Editor	
PL-26516	Improved the PCF editor to highlight the correct panel when selecting widgets in nested files.
PL-27147	The <code>PCFMapping</code> tool has been updated to include fields such as <code>PanelIterator</code> .
IntelliJ IDE – IntelliJ IDE – Plugins, OSGi, Plugins	
PL-27497	You can now implement plugin interfaces in Java using the OSGi standard. OSGi is a Java module system and service platform that helps isolate code modules and any necessary Java libraries. Guidewire recommends OSGi for all new Java plugin development. To simplify OSGi configuration, ContactManager includes IntelliJ IDEA with OSGi Editor, an application separate from Guidewire Studio. For more information, refer to the <i>Integration Guide</i> .
IntelliJ IDE – Refresh	
PL-24108	Fixed an issue in which the server threw an exception and did not handle newly created enhancements.
PL-28174	Fixed an issue where methods added in entity classes were invalid until restarting Studio.
PL-28187	Fixed an error that would occur after renaming an element in an XSD and then navigating to a Gosu type that contained a usage of that element.

IntelliJ IDE – Typelist Editor

PL-24391	Fixed the typelist editor in Studio to filter out options under the drop-down as you enter text.
PL-26445	Fixed an issue in which creating the first extension of a typelist caused an exception.
PL-26613	Fixed an issue in Studio in which disabled typecodes looked enabled when selected.
PL-26535	Fixed an exception in the text editor of the typelist view.
PL-27174	Fixed an issue in which the default setting for filtering metadata did not apply to the Typelist editor.
PL-27570	Fixed an issue in which clicking on the name attribute in a typelist extension would not allow you to override it.
PL-27584	Fixed the Add To Category dialog so retired typecodes appear in strikethrough text as options for filtering.
PL-28177	In Studio, the Entity editor and Typelist editor are now case-insensitive when resolving references to other metadata.
PL-28515	Fixed an issue that generated multiple errors in the Studio Typelist editor in the localization panel.

IntelliJ IDE – Web Services Editor

PL-26576	Added a check to the timeout value of web services to insure that it is lower than Studio's maximum integer.
----------	--

Localization

CTC-1893	When you are editing a contact who is a Person subtype in ClaimCenter or an ABPerson subtype in ContactManager, there is a Driver's License section on the edit screen. The State field on this screen now uses the <code>Jurisdiction.ttx</code> file to populate the drop-down list for the field. If you want to change the contents of the drop-down list, edit the <code>Jurisdiction.ttx</code> file in Studio and add or remove values from the outgoing category <code>driving_lic</code> . Be sure to make this change in both ClaimCenter and ContactManager.
CTC-1936	Kanji fields require a case-sensitive search. Previously, in the base configuration, search for Kanji contact fields was using a case-insensitive search. Now in the base configuration, searching for these fields is case-sensitive. Notes: <ul style="list-style-type: none"> With Kanji fields, use <code>startsWithCaseSensitive</code> in <code>search-config.xml</code> to do a case-sensitive search. If you use <code>startsWith</code>, the search is case-insensitive. You can use <code>eq</code> in <code>search-config.xml</code> to do a case-sensitive search if the referenced field has <code>supportsLinguisticSearch</code> set to <code>true</code> in the schema. Otherwise, <code>eq</code> performs a case-insensitive search.
CTC-2104	The minimum set of fields required for a contact search can vary by country and locale. The error message displayed in response to an invalid contact search now varies based on the country specified in the search and the locale setting of the user making the search.
CTC-2238	The <code>DefaultApplicationLanguage</code> configuration parameter is now visible in the <code>config.xml</code> file.
CTC-2332	Previously, the <code>TaxID</code> field was required when creating a <code>VendorPerson</code> or <code>VendorCompany</code> contact and any subtypes of those entity types, even for locales that did not require this field to create a vendor. This problem has been fixed. The PCF files no longer require this field. Additionally, you can now set whether <code>TaxID</code> is required in the plugin implementation class <code>ValidateABContactCreationPluginImpl</code> , as described in the Contact Management Guide.
CTC-2335	In the search results for a contact search, the Country search criterion now determines the labels for the State and PostalCode fields of the Address entity. For example: <ul style="list-style-type: none"> US – State and ZIP Code Japan – Prefecture and Postcode Canada – Province and Postal Code Most others – State and Postcode In ContactManager, the values of these field labels are defined in <code>ContactSearchResultsLV.pcf</code> . In ClaimCenter, the values of these field labels are defined in <code>AddressBookSearchLV.pcf</code> . In both applications, the two field labels are populated by the following code: <ul style="list-style-type: none"> <code>gw.api.address.AddressCountrySettings.getSettings(searchCriteria.Address.Country).StateLabel</code> <code>gw.api.address.AddressCountrySettings.getSettings(searchCriteria.Address.Country).PostalCodeLabel</code>

Other – Persistence

PL-25820 Fixed an issue with whitespace not being trimmed by trimming unicode full-width whitespace.

Profiling

PL-28172 Fixed an issue in which the Purge Profiler Data batch job would throw exceptions when purging web services profiling data.

Queries

PL-18578 This release adds a new class, `gw.api.database.QuerySelectColumns`, with static methods that help you specify the columns that you want selected in a row query. Instead of passing a Gosu block to the select method, you pass a list of `IQuerySelectColumn` objects, which you construct by using static methods on the `QuerySelectColumns` class. Each `IQuerySelectColumn` object represents a column in the result. For example, the following Gosu sample code creates a row query, with `Address` as the primary entity. The result includes only the `City` column for `Address` instances that match the query criteria.

```
uses gw.api.database.Query
uses gw.api.database.QuerySelectColumns
uses gw.api.path.Paths
var addressQuery = Query.make(Address)
...
// Join and condition statements go here.
...
var addressResult =
    addressQuery.select({QuerySelectColumns.path(Paths.make(Address#City))})
```

The `QuerySelectColumns` class includes static methods that represent database functions to help you construct aggregate queries. For example, the following Gosu code returns a count of all `Address` instances.

```
uses gw.api.database.DBFunction
uses gw.api.database.Query
uses gw.api.database.QuerySelectColumns
uses gw.api.path.Paths
var addressQuery = Query.make(Address).
var addressQuery.withDistinct(true) // Always run aggregate queries
                                   with Distinct set to true.
addressResult = addressQuery.select({QuerySelectColumns.dbFunction(
    DBFunction.Count(Paths.make(Address#City))
})})
```

PL-27474 Fixed an issue in which the `Count` property on a `gw.api.database.Query` result was producing an incorrect SQL statement and count result when the `withDistinct(true)` setting was combined with a `Join`.

User Interface

CTC-2047 In the ContactManager user interface, the contact history list view now displays international phone numbers correctly.

CTC-2235 In the ContactManager user interface, `PersonVendor` detail views are now showing the cell phone field.

Web - ListViews

PL-10908 Added the `groupedOnEnter` attribute to many cell-based PCF elements. If true, the `ListView` is grouped by this cell upon entering the page. Only one grouped cell is allowed at any given time, and it is applicable only when the column is sortable.

PL-27732 A new `height` attribute has been added to the `ListViewPanel` PCF element. This attribute sets the vertical size in pixels of the list view. If the list data is taller than the specified height, a vertical scroll bar appears within the list view. The height is calculated from the list view header toolbar. Header rows are fixed, and the footer scrolls with the data. Note that this attribute is currently experimental and might not function properly. You should fully test any use of this attribute.

PL-28466 Fixed an issue with columns reordering when switching between filtering views in a list view.

Web – UI/Runtime

PL-28749	Fixed a critical security vulnerability to persistent cross-site scripting attacks.
PL-27884	The following new methods were added to the Javascript <code>gw.api.Util</code> class: <code>getValue</code> , <code>setValue</code> , <code>getValues</code> , <code>setValues</code> . These methods enable you to get or set values of input elements. For example, you can use these methods in <code>TemplatePanel</code> and pass Gosu variables as arguments into these methods.
PL-28336	Fixed an issue that would occur when there was an action defined on a range cell.
Work Queues	
PL-27301	Improved server performance when selecting available work queue items.
PL-28696	Fixed an issue with <code>workItem</code> orphan detection during daylight saving time adjustment.
XMLElement (and XSD types)	
PL-27131	Gosu XML/XSD types now support use of circular <code>xs:include</code> references.

Known issues and limitations for ContactManager 8.0.1

This topic describes known issues with this release of Guidewire ContactManager.

Note: For maintenance releases, Guidewire often defers fixing configuration issues that require merging files during the upgrade. Workarounds to many of these issues are listed in the following topics. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 8.0.1 known issues

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not verify that the relationship does not already exist, and therefore can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one `ContactContact` entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the `RelatedABContactSearchCriteria` entity and the `CriteriaDef` elements that are defined for it in `search-config.xml` is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

Proximity search integration between ClaimCenter 7 and ContactManager 8 is broken (CTC-2443)

Issue – Contact proximity search from ClaimCenter 7.0 to ContactManager 8.0 does not work. There are two fields missing in the ContactManager 8.0 WSDL that are required by ClaimCenter 7.0 contact proximity search.

Workaround – In Guidewire Studio™ for ContactManager 8.0, add the following two fields to the `ab700` version of the `ProximitySearchParametersInfo` class:

```
public var RadiusSearchMaxResults: Integer
public var GUID : String
```

This class is in the package `gw.webservice.ab.ab700.abcontactapi`.

After adding these two fields to the ContactManager class, regenerate the ContactManager 8.0 WSDL and refresh the WSDL in ClaimCenter 7.0, as follows:

1. Restart the ContactManager 8.0 server to regenerate the WSDL.
2. Open Guidewire Studio™ for ClaimCenter 7.0.
3. In the **Resources** pane on the left, navigate to **configuration→Classes→wsi→remote→gw→webservice→ab→ab700**.
4. In the editor on the right, in the **Resources** pane, select the following resource:

```
${ab}/ws/gw/webservice/ab/ab700/abcontactapi/ABContactAPI?wsdl
```

5. At the bottom of the pane, click **Fetch Updates**. Fetching updates for ABContactAPI also pulls in changes to the ContactManager Search web services.
6. Restart the ClaimCenter server.

Platform/Studio known issues for ContactManager 8.0.1

Problem with regen-java-api command and JAR files (PL-16351)

Issue – If you run the `gwab regen-java-api` command, ContactManager creates a *ContactManager/java-api/lib* directory and puts JAR files for Java development in this directory. However, ContactManager does not always perform this task correctly, resulting in some of the generated JAR files not being copied to this directory.

Workaround – For missing library files, copy or import into your Java development environment the missing library files from the following directory:

```
ContactManager/admin/lib
```

Copy them into the following directory:

```
ContactManager/java-api/lib
```

Renaming method or property throws ParseResultsException (PL-16633)

Issue – If you rename a property or a method or you change a method signature, and a workflow references that property or method in a Gosu field, ContactManager throws `ParseResultsException`. This is the intended behavior.

Workaround – Restart the workflow engine. To do so:

1. Log in to ContactManager using an administrative account.
2. Access **Internal Tools→Reload**.
3. Click **Reload Workflow Engine**.

Javadoc command does not generate index file in expected location (PL-27679)

Issue – When you run `gwab regen-java-api`, an `index.html` file is not created in *ContactManager/java-api/doc*.

Workaround – The command now generates Javadoc JAR files in *ContactManager/java-api/doc*. To view the Javadoc, add the JAR files to the Studio project.

Gosu does not automatically downcast if the left side of the `typeis` or `typeof` expression uses deprecated members (PL-27724)

Issue – To improve readability of your Gosu code, Gosu automatically downcasts after a `typeis` expression if the type is a subtype of the original type. This is particularly valuable for `if` statements and similar Gosu structures. For example, if a variable has type `Object`, you can use code such as:

```
if( x typeis String ) {
    length // NOTE: length is a property on String, but *not* on Object.
}
```

In this release, Gosu does not automatically downcast if the left side of the `typeis` or `typeof` expression uses deprecated members. This may result in new compilation errors.

Workaround – To fix these compiler errors, explicitly downcast with the `as` keyword before you access properties or methods on the subtype but not on the original type. For example, suppose a property called `Dep` is deprecated:

```
if (x.Dep typeis ExampleType) {  
    return (x.Dep as ExampleType).PropertyOnExampleSubtype  
}
```

Long text in table cells can add white space to the right of the page (PL-28288)

Issue – In an editable list view, extremely long text entered in a single cell can cause additional white space on the right side of the page. Long text is text that occupies approximately the entire width of the screen. This issue occurs primarily in Chrome.

Workaround – If you expect users to enter large amounts of text into the cells of a column, configure the column to support text wrapping.

AddressZoneValue in zone-config.xml files uses display name instead of code (PL-28511)

Issue – Some of the `<AddressZoneValue>` expressions in the US, AU, CA, and DE `zone-config.xml` files use the `DisplayName`, rather than the `Code`, of the associated `State.Abbreviation`. Configuring `ContactManager` to use translated names for the `StateAbbreviation` typelist can result in failure to find some zones for that country.

Workaround – Replace references to `State.Abbreviation.DisplayName` with `State.Abbreviation.Code` in `<AddressZoneValue>` elements.

In each `zone-config.xml` file, replace the following `DisplayName` property:

```
<AddressZoneValue>Address.State.Abbreviation.DisplayName</AddressZoneValue>
```

Instead, use the following `Code` property:

```
<AddressZoneValue>Address.State.Abbreviation.Code</AddressZoneValue>
```

Problem with running regen-java-api command with deprecated=true flag (PL-28992)

Issue – If you run the `gwab regen-java-api` command with the flag `-Ddeprecated=true`, the command can fail.

Workaround – Specify `dev-deploy` as the value of the `depends` attribute, as follows:

1. Open the `build.xml` file in `ContactManager/modules/ant` in an editor.
2. Find the entry that starts as follows:

```
<target name="regen-java-api" depends="init"
```

3. Change the value of `depends` to `dev-deploy`, as follows:

```
<target name="regen-java-api" depends="dev-deploy"
```

4. Save the file and run the `regen-java-api` command.

Administrative command-line tools cannot refresh WSDL (PL-29021)

Issue – Some administrative command-line tools rely on web service implementation classes such as `MaintenanceToolsAPI.gs`. Source files for these classes use the `@Export` annotation, which allows you to edit the file. In this release, the administrative command-line tools cannot refresh the WSDL for these classes. Any change to the web service implementation class that changes the WSDL can prevent the administrative tools from working. Therefore, the only changes you can make to these classes are changes that do not affect the WSDL. For example, you can add `@WsPermission` annotations to change the permissions without changing the WSDL.

The web services used in the command-line tools for ContactManager are ImportToolsAPI, MaintenanceToolsAPI, MessagingToolsAPI, SystemToolsAPI, TableImportAPI, WorkflowAPI, and ZoneImportAPI.

Workaround – Guidewire is aware of this issue.

Security Vulnerability - Reflected XSS (PL-29052)

Issue – There is a non-persistent cross-site scripting (reflected XSS) vulnerability. Unlike other XSS types, this vulnerability does not permit privilege escalation and does not propagate easily to other users.

Workaround – Guidewire is aware of the issue. Strong email filtering with phishing/malware detection is an effective defense against this type of attack. Contact Customer Support for more information.

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, you can view the HTML Guidewire product documentation only if it is served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

New mechanism for reloading Gosu classes (DOC-8218)

Issue – In past releases, you could modify your PCF files and Gosu classes in Studio, and then reload the changes into the running server by pressing `Alt+Shift+L` in the application user interface. This shortcut no longer loads Gosu classes.

Workaround – To have the server reload your Gosu classes, in Studio, click **Build→Make Project**. When Studio is finished compiling your project, the changes will be loaded. You can also restart your server to load the Gosu classes.

Guidewire ContactManager 8.0.0 release notes

These release notes contain the following sections:

- “Release Information for ContactManager 8.0.0” on page 400
- “Installing ContactManager 8.0.0” on page 401
- “Support” on page 21
- “Issues and major changes for ContactManager 8.0.0” on page 401
- “Known issues and limitations for ContactManager 8.0.0” on page 402

Release Information for ContactManager 8.0.0

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you are upgrading and you skipped one or more releases of ContactManager, be sure to read the release notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 8.0.0” on page 401.

This release of Guidewire ContactManager is 8.0.0.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later. The latest maintenance release is preferred.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 7.0.0 or later. The latest maintenance release is preferred.

Installing ContactManager 8.0.0

For general installation information, see “Installing ContactManager” on page 55.

For versions of ContactManager prior to 8.0.0 that you have skipped, see:

- “Guidewire ContactManager 7.0.7 release notes” on page 405
- “Guidewire ContactManager 7.0.6 release notes” on page 412
- “Guidewire ContactManager 7.0.5 release notes” on page 416
- “Guidewire ContactManager 7.0.4 release notes” on page 423
- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include sections for ContactCenter release notes. For example, see the release note archive in the *New and Changed Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Issues and major changes for ContactManager 8.0.0

This topic contains issues and major changes that might affect your installation.

- “Base PCF file changes for ContactManager 8.0.0” on page 401
- “Rules changes for ContactManager 8.0.0” on page 401
- “Changes in this release provided in Upgrade Diff report” on page 311

Base PCF file changes for ContactManager 8.0.0

The link that follows requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

ContactManager release 7.0.5 to 8.0.0

To view a report of the changes to the base PCF files, refer to the original ContactManager 8.0.0 release notes.

Rules changes for ContactManager 8.0.0

The link that follows requires that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

ContactManager release 7.0.5 to 8.0.0

To view a report of the changes to the base rules, refer to the original ContactManager 8.0.0 release notes.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Known issues and limitations for ContactManager 8.0.0

This section describes known issues with this release of Guidewire ContactManager:

- “ContactManager known issues for release 8.0.0” on page 402
- “Platform/Studio known issues for ContactManager 8.0.0” on page 403

ContactManager known issues for release 8.0.0

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one ContactContact entity. ContactManager does not verify that the relationship does not already exist. Therefore ContactManager can create multiple, redundant ContactContact entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configuration of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one ContactContact entity in this case.

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the RelatedABContactSearchCriteria entity and the CriteriaDef elements that are defined for it in search-config.xml is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

ClaimCenter ignoring ContactMapper persist properties when a contact is initially linked (CTC-2112)

Issue – Setting .withPersist(false) for the field of a Contact entity or subentity in the ContactMapper class in ClaimCenter does not always have the intended effect. It is supposed to prevent the field from being persisted in the ClaimCenter database. However, in the following case, ClaimCenter is ignoring this setting. When you initially add an existing contact from ContactManager to ClaimCenter, all properties on the contact brought over from ContactManager are saved in the ClaimCenter database.

Updates from ContactManager for subsequent sync operations on the linked contact do work correctly. These subsequent sync operations do not persist properties that have been marked .withPersist(false).

Workaround – Guidewire is aware of this issue.

The DefaultApplicationLanguage property is missing from the config.xml file (CTC-2238)

Issue – In the base configuration, the configuration file config.xml is missing the DefaultApplicationLanguage property. This issue is a problem only because developers expect to see this parameter in the file and might not think to set it if it is not there. There are other parameters that are not in config.xml that can be set simply by adding and setting them, as needed, so a missing parameter is not necessarily a problem.

Workaround – Add the property to config.xml and set it to the default language. During installation, and before starting the database for the first time, you must provide a value for this configuration parameter. This value must be a typecode that exists in the LanguageType typelist.

Platform/Studio known issues for ContactManager 8.0.0

Chrome browser cannot display product documentation in HTML format (DOC-7251)

Issue – If you use the Google Chrome browser, to view the HTML Guidewire product documentation, it must be served by an HTTP server using the `http://` protocol. The Chrome browser cannot load HTML product documentation from your local disk by using the `file://` protocol.

Workaround – Use a different browser, such as Microsoft Internet Explorer. If you use the HTML documentation only to access the PDF files, you can open them directly in the `pdf` subdirectory of the `doc` directory.

GX model generated XSD cannot be parsed by JAXB (PL-13598)

Issue – XSD generated by the GX model cannot be parsed by JAXB.

Workaround – Add JAXB annotation elements to the XSD to specify the necessary metadata, such as class names, to enable JAXB to generate the Java class files. Contact Guidewire Support for a sample XSD file that is annotated this way.

Problem with regen-java-api command and JAR files (PL-16351)

Issue – If you run the `gwab regen-java-api` command, ContactManager creates a `ContactManager/java-api/lib` directory and puts JAR files for Java development in this directory. However, ContactManager does not always perform this task correctly, resulting in some of the generated JAR files not being copied to this directory.

Workaround – For missing library files, copy or import into your Java development environment the missing library files from the following directory:

```
ContactManager/admin/lib
```

Copy them into the following directory:

```
ContactManager/java-api/lib
```

Renaming method or property throws ParseResultsException (PL-16633)

Issue – If you rename a property or a method or change a method signature, and a workflow references that property or method in a Gosu field, ContactManager throws `ParseResultsException`. This is the intended behavior.

Workaround – Restart the workflow engine. To do so:

1. Log in to ContactManager using an administrative account.
2. Access **Internal Tools**→**Reload**.
3. Click **Reload Workflow Engine**.

Gosu class can override @InternalAPI methods when a PublishInGosu java class is subclassed (PL-18217)

Issue – When a Gosu class extends a Java class, it is possible for the Gosu class to override methods in the Java class that are marked as `@InternalAPI`. These overrides could lead to unpredictable behavior.

Workaround – Do not override methods marked as `@InternalAPI` when creating a Gosu class that subclasses a Java class.

Client-side document production scripts cannot be customized in this release (PL-21502)

Issue – In previous releases, you could customize client-side document production scripts downloaded by the ActiveX Document Assistant. You could modify JScript files in the web application and remove the cached copies from a `temp` directory on all user computers. In this release, the ActiveX control was replaced by a signed Java Web Start (JWS) application. Because client-side scripts are encapsulated in the signed JWS application, you cannot change the scripts in this release.

Workaround – Guidewire is aware of this issue.

New inbound integration system requires additional configuration information (PL-25227)

Issue – This release includes a new inbound integration system, which is documented in the ContactManager Integration Guide. Additional configuration information is necessary to use the new API in this release.

Workaround – Contact Guidewire Customer Support for details.

JBoss 6 application server unable to start (PL-27203)

Issue – JBoss 6 generates an exception when it is deployed with ContactManager 8.0.0.

Workaround – Remove or comment out the tag `<resource-ref>` in the file `web.xml`.

Multiple rule folders are created during a configuration upgrade (PL-27338)

Issue – Multiple rule folders are created if you repeatedly run the configuration upgrade tool followed by the `clean` command.

Workaround – Restore the innermost rules folder to its proper location.

1. Copy the innermost folder to a temporary location.
2. Remove all the nested folders from the original location.
3. Copy back the innermost folder from the temporary location to its proper location.
4. Make sure the folder is named correctly (`ContactManager8_0_0Rules`).

List view columns that are initially not visible and then set to visible always appear on the right side (PL-27556)

Issue – In ContactManager 8.0, you can reorder the columns of list views or change their width. These settings are then saved by ContactManager in the web browser for each list view, so the same ordering and width can be used when the page is revisited.

However, a layout of a list view can change due to differences in data or because the server configuration has changed. New columns added to the list view since the last time the user visited show up on the far right side of the list view. They are not in the order specified in the PCF file. This can be confusing, especially when you must scroll the page to the right to see the new columns.

This behavior can also occur when there are two modes of a PCF page containing list views of a similar structure, but with a different ordering of columns. The order and width settings can be applied to the wrong list view in this case, and columns can appear in a different order than intended.

Workaround – If list view columns seem to be missing, first scroll to the right to see if they are there. To correct the order of list view columns, you can reset your layout preferences to restore the default list ordering and widths. To do this, select **Options** → **Clear Layout Preference** in Guidewire ContactManager.

Javadoc command does not generate index file in expected location (PL-27679)

Issue – When you run `gwab regen-java-api`, an `index.html` file is not created in `ContactManager/java-api/doc`.

Workaround – The command now generates Javadoc JAR files in `ContactManager/java-api/doc`. To view the Javadoc, add the JAR files to the Studio project.

Command to generate data dictionary fails if maxSPVInclusions option is specified (PL-27693)

Issue – The data dictionary is not generated when you run the `gwab regen-dictionary` command with the `maxSPVInclusions` option.

Workaround – Do not use the `maxSPVInclusions` option with this command.

Gosu does not automatically downcast if the left side of the `typeis` or `typeof` expression uses deprecated members (PL-27724)

Issue – To improve readability of your Gosu code, Gosu automatically downcasts after a `typeis` expression if the type is a subtype of the original type. This is particularly valuable for `if` statements and similar Gosu structures. For example, if a variable has type `Object`, you can use code such as:

```
if( x typeis String ) {
    length // NOTE: length is a property on String, but *not* on Object.
}
```

In this release, Gosu does not automatically downcast if the left side of the `typeis` or `typeof` expression uses deprecated members. This may result in new compilation errors.

Workaround – To fix these compiler errors, explicitly downcast with the “`as`” keyword before you access properties or methods on the subtype but not on the original type. For example, suppose a property called `Dep` is deprecated:

```
if (x.Dep typeis ExampleType) {
    return (x.Dep as ExampleType).PropertyOnExampleSubtype
}
```

Upgrade trigger for `postOnChange` on PCF widgets is not working in some cases (PL-27755)

Issue – In PCF files for some widgets, the ContactManager 8.0.0 upgrade tool does not upgrade the `postOnChange` property to the new syntax for this property. This problem can occur with any widgets that you have added that use `postOnChange` and with any widgets in the base configuration for which you have set `postOnChange`.

Workaround – After running the ContactManager 8.0.0 upgrade tool:

1. Find all instances of widgets that did not have their `postOnChange` properties converted. For example, search the `ContactManager/modules/configuration/config/web/pcf` folder and subfolders for occurrences of `postOnChange=`. The instances that you will find are either widgets that need correction or widgets that are commented out (disabled.) There is no need to make the correction on disabled widgets, although there is also no harm in doing so.
2. For each widget that needs correction, open its PCF file in an XML editor and change the following old syntax to the new syntax:
 - Old syntax example:
3. To verify that you have corrected all instances, open Guidewire Studio and navigate in the **Project** window to **configuration**→**config**→**Page Configuration**. Then compile all files in the `pcf` folder. If there are no errors relating to `postOnChange`, your corrections are complete.

New mechanism for reloading Gosu classes (DOC-8218)

Issue – In past releases, you could reload your PCF files and Gosu classes in Studio by using a key combination. To reload the changes into the running server, you pressed `Alt+Shift+L` in the application user interface. This shortcut no longer loads Gosu classes.

Workaround – To have the server reload your Gosu classes, in Studio, click **Build**→**Make Project**. When Studio is finished compiling your project, the changes will be loaded. You can also restart your server to load the Gosu classes.

Guidewire ContactManager 7.0.7 release notes

These release notes contain the following:

- “Release Information for ContactManager 7.0.7” on page 405
- “Installing ContactManager 7.0.7” on page 406
- “Support” on page 21
- “Major issues and changes for ContactManager 7.0.7” on page 406
- “Improvements and resolved issues for ContactManager 7.0.7” on page 407
- “Known issues and limitations for ContactManager 7.0.7” on page 409

Release Information for ContactManager 7.0.7

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you are upgrading and you skipped one or more releases of ContactManager, be sure to read the release notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 7.0.7” on page 406.

This release of Guidewire ContactManager is 7.0.7.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 6.0.7 or later.

Installing ContactManager 7.0.7

For general installation information, see “Installing ContactManager” in the ClaimCenter Contact Management Guide.

For versions of ContactManager prior to 7.0.7 that you have skipped, see:

- “Guidewire ContactManager 7.0.6 release notes” on page 412
- “Guidewire ContactManager 7.0.5 release notes” on page 416
- “Guidewire ContactManager 7.0.4 release notes” on page 423
- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include sections for ContactCenter release notes.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Major issues and changes for ContactManager 7.0.7

This topic contains issues and major changes that might affect your installation.

Allowing Core Applications to Specify the LinkId for a New Contact (CTC-3078)

ContactManager 7.0.7 now allows a core application to suggest a value for the LinkId property when requesting creation of a contact. Enabling this feature requires configuration of both the core application and ContactManager.

After this configuration:

- The core application can supply a value for External_Unique_ID in the Contact XML to specify a unique value for LinkID. It is up to the application to ensure that this value is unique. The core application generates and maps this value in the XML sent to ContactManager.
- When ContactManager detects a value for External_Unique_ID in the XML received from a core application, ContactManager uses that value for LinkID instead of generating one. If that LinkID already exists for a Contact, ContactManager throws a DuplicateKeyException.

Note: PolicyCenter 8.0.1 and later requires that ContactManager be able to handle such a request.

See also

- “Configuring ContactManager 7 to Support LinkID Requests from Core Applications” in the *Contact Management Guide*

Internet Explorer 11 is now supported (PL-30386)

All releases of ContactManager 7 now support Internet Explorer 11. To use Internet Explorer 11, you must enable quirks mode on your web server. For more information, visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

Base PCF file changes in ContactManager 7.0.7

The following link requires that the ReleaseNotes_files directory be on your local disk in the same directory as this release notes file.

ContactManager release 7.0.6 to 7.0.7

- There are no changes to the base PCF files in the modules/ab directory.
- To view a report of the changes in the base PCF files in the modules/p1 directory, refer to the original ContactManager 7.0.7 release notes.

Base rule changes in ContactManager 7.0.7

ContactManager release 7.0.6 to 7.0.7

- There are no changes to the base rules.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and resolved issues for ContactManager 7.0.7

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ContactManager 7.0.7 improvements and resolved issues

ID	Description
Contact Domain	
CTC-3098	An issue has been fixed that could occur during database upgrade from ContactCenter 5 or 6 to ContactManager 7. ContactManager no longer attempts to create tags for ABContact objects that are retired in the database.
Miscellaneous	
CTC-3103	An issue that could occur during upgrade concerning the length of generated LinkID values for ABContactTag objects has been fixed. The links generated for these objects no longer exceed the length of the data type for LinkID fields.

Platform improvements and resolved issues for ContactManager 7.0.7

ID	Description
Bundles and Transactions	
PL-31943	Fixed an issue where parameters in debugging information for failed SQL statements did not match the variables in the SQL statements.
Clustering	
PL-29170	<p>Added a JGroups watchdog timer that resets cluster nodes if they stop responding to the cluster. This includes the following new configuration parameters:</p> <ul style="list-style-type: none"> JGroupsWatchdogHeartbeatIntervalSecs - The number of seconds between each heartbeat ping from the cluster coordinator to each of the nodes. Default: 30 seconds JGroupsWatchdogMissedHeartbeatsBeforeReset - The number of missed heartbeats after which a node resets its JGroups channel. Default: 10
PL-29237	Changed the default value for the ClusterProtocolStack configuration parameter. For details, see the <i>Configuration Guide</i> .
Command Line Tools	
PL-30887	The command <code>gwab verify-resources</code> now correctly passes the <code>reportingPeriod</code> parameter.
Configuration Upgrade	
PL-32616	In the Configuration Upgrade tool, removed the right-click pop-up menu.
Database Instrumentation	
PL-30150	<p>The following options, which are selected by default in the base configuration, have been added to the Oracle AWR info page. You can also deselect any of these options when generating an Oracle AWR report:</p> <ul style="list-style-type: none"> Analysis of messaging Analysis of batch processing and work item Analysis of history
Database Support - SQL Server	
PL-31638	<p>The database upgrade converts SQL Server primary keys from an integer to a long integer (bigint) and datetime columns to datetime2 columns when upgrading from a version prior to 7.0. The performance of this upgrade step has been improved. For maximum performance, set the <code>allowUnloggedOperations</code> upgrade attribute in <code>config.xml</code> to <code>true</code> to minimize logging.</p> <pre><database ...> ... <upgrade allowUnloggedOperations="true"> ... </upgrade> </database></pre> <p>This option requires that the recovery model is bulk or simple, and that data and transaction replication are off. See “Enabling Migration to 64-bit IDs (SQL Server Only)” in the <i>Upgrade Guide</i>.</p>
Database Upgrade	
PL-32092	Resolved an issue that could cause an upgrade to fail on SQL Server if running the upgrade with the parameter <code>MigrateToLargeIDsAndDatetime2</code> set to <code>true</code> .
Gosu	
PL-29166	Fixed an error that would occur during a type system refresh.
Messaging	
PL-23930	Fixed a thread safety issue in <code>com.guidewire.pl.system.integration.messaging.schema.DestinationNode.getPositionOfEventName</code> that would cause poor performance.

ID	Description
PL-29999	Eliminated unnecessary locking and unlocking in the MessageRequest plugin when it is not configured.
Other - Cloud	
PL-31929	CSV export from a list view now correctly handles newline characters (CR, LF) in the source data.
PL-32042	Added support for recent time zone changes, such as those in Russia.
Other - Persistence Pod	
PL-31177	Fixed an issue that caused the exception "DB Exception - database attempted to divide by 0" when performing a calculation on a date range that contained a leap day.
Queries	
PL-17270	Fixed an issue in which <code>gw.api.database.QueryProcessor.Count</code> was producing an incorrect SQL statement and count result.
Revisioning	
PL-28241	Added a comment with the source of a lock to the <code>SELECT FOR UPDATE</code> statement. This gives a useful context when debugging locking issues, including being able to separate out the calls from messaging rules from the calls from within messaging.
Search	
PL-32334	Added new Boolean parameter <code>securetransport</code> to <code>solrserver-config.xml</code> for <code>http</code> and <code>cluster</code> server types. When set to <code>true</code> , HTTPS is used instead of HTTP for Solr server connections. Secure transport is not supported for embedded operation. Solr server certificates must be added to the JDK or application server trust store. SSL must be enabled on the server hosting Solr.
Web - Other	
PL-21303	The HTTP <code>Cache-Control</code> header is now set to "no-store, no-cache".
PL-29418	Fixed a problem that prevented the Admin page from being shown after applying WebSphere patch <code>7.0.0.0-WASJavaSDK-LinuxX64-IFPM98578.pak</code> .
Web - UI/Runtime	
PL-31652	Fixed an issue where some HTML in a <code>TreeView</code> was not properly escaped.
Web Services - WSI (New)	
PL-30758	Fixed a bug in WSDL generation for services with multiple <code>@Throws</code> annotations for the same exception but different reasons. The bug caused an exception in some cases for WSDL consumers. The WSDL now merges related faults into one fault with all explanations concatenated in one comment. The fix prevents an exception that some WSDL consumers experienced.
PL-32545	Fixed a security vulnerability to XML external entity (XXE) attacks. For more information about this issue, or to learn about temporary workarounds while you deploy this fix, visit the Guidewire Community and search for knowledge article 7153, "Are Guidewire PolicyCenter, BillingCenter, ClaimCenter, and ContactManager vulnerable to XML external entity (XXE) attacks in their web service layers?"
Work Queues	
PL-31572	You can now create custom distributed work queues by developing Gosu classes that subclass from the <code>WorkQueueBase</code> class. For more information, see the Javadoc and Gosudoc.
XMLElement (and XSD types)	
PL-30792	Fixed a bug in WSDL import relating to handling simple data if a <code>complexType</code> is a <code>complexContent</code> extension of a <code>complexType</code> with a <code>simpleContent</code> extension.

Known issues and limitations for ContactManager 7.0.7

This topic describes known issues with this release of Guidewire ContactManager.

Note: For maintenance releases, Guidewire often defers fixing configuration issues if they require merging files during the upgrade. Workarounds to many of these issues are listed in the following sections. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 7.0.7 known issues

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the `RelatedABContactSearchCriteria` entity and the `CriteriaDef` elements that are defined for it in `search-config.xml` is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not check to make sure the relationship does not already exist, and therefore can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configurations of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one `ContactContact` entity in this case.

Email field validation in ContactManager not the same as in core applications (CTC-1339)

Issue – The entry in `fieldvalidators.xml` in ContactManager is more restrictive than in the core applications. This difference can cause issues while creating a contact in a core application.

Workaround – Change the ContactManager version of this validator to match the validators in the core products. Use Guidewire Studio™ to edit the entry in `fieldvalidators.xml` for `Validator.Email`, changing its value from the default `".+@.+\..+"` to the value in the core applications, `".+@.+"`.

Studio/Platform known issues for ContactManager 7.0.7

Problem with regen-java-api command and JAR files (PL-16351)

Issue – If you run the `ContactManager/bin/gwpc regen-java-api` command, ContactManager creates a `ContactManager/java-api/lib` directory and puts JAR files for Java development in this directory. However, ContactManager does not always perform this task correctly, resulting in some of the generated JAR files not being copied to this directory.

Workaround – For missing library files, copy or import into your Java development environment the missing library files from the following directory:

```
ContactManager/admin/lib
```

Copy them into the following directory:

```
ContactManager/java-api/lib
```

Renaming method or property throws ParseResultsException (PL-16633)

Issue – If you rename a property or a method or you change a method signature, and a workflow references that property or method in a Gosu field, ContactManager throws `ParseResultsException`. This behavior is intended.

Workaround – Restart the workflow engine. To do so:

1. Log into ContactManager using an administrative account.
2. Access **Internal Tools**→**Reload**.
3. Click **Reload Workflow Engine**.

In some languages, web browsers render column headers of list views improperly (PL-18027)

Issue – In some languages, web browsers render some column headers of list views improperly if their column widths are specified too narrowly in their PCF definitions. For example, sometimes a numeric column is specified with a variable width of 1%. This narrow setting forces the browser to render the column too narrowly for the text of the translated column heading.

Workaround – Edit the PCF file that defines the column and clear the value from the width property. Without a specified value for the column width, browsers render the column widely enough to display the full text of the translated column heading.

Database upgrade does not handle nullable to non-nullable columns with a default value for subtypes (PL-23104)

Issue – For entity definitions, the automatic database upgrade converts nullable columns to non-nullable with a default value successfully. However, this column type conversion is not possible for columns in subtype definitions. ContactManager implements non-nullable columns on subtype in the database as nullable because that column must have null values for rows that represents instances of other subtypes.

Workaround – Write a version trigger to populate the column with the default value for existing rows for the subtype. After you upgrade, ContactManager enforces the column value to be non-nullable with the default value for new rows of the subtype.

New transport plugin definitions do not show in the list of valid transport plugins (PL-23317)

Issue – Newly implemented transport plugin definitions do not show in the list of valid transport plugins displayed by clicking the light-bulb icon next to the Transport Plugin field in the messaging destination editor.

Workaround – Restart Studio.

Find in Resources fails for resources under Data Model Extensions or Web Resources (PL-23320)

Issue – In Studio, the **Find in Resources** option does not work for resources that are in Data Model Extensions or Web Resources.

Workaround – To see the full resource path, hover over the resource name in the tab of the resource editor. Then, navigate in the resource pane on the left to find the resource in the resource hierarchy.

No entity changed event fired through a one-to-one entity relationship (PL-26224)

Issue – The existence of a one-to-one entity relationship between two entities prevents an entity change event from being fired, even if it is supposed to. For example, in ContactManager there is a foreign key from ABContact to Address for the primary address, and a one-to-one relationship from Address to ABContact. Because of the one-to-one link, no event changed event is fired for any change in the primary address.

More generally, if entity E1 has a one-to-one relationship to entity E2, no entity changed event is fired for E2 if a property on E1 changes.

Workaround – You might be able to take advantage of a secondary change caused by the initial change, with the secondary change triggering the event to be fired instead. In the previous ContactManager example, making sure that changes to all properties in Address cause a History record to be generated ensures that the entity changed event will be fired.

Guidewire ContactManager 7.0.6 release notes

These release notes contain the following:

- “Release Information for ContactManager 7.0.6” on page 412
- “Installing ContactManager 7.0.6” on page 412
- “Support” on page 21
- “Issues and major changes for ContactManager 7.0.6” on page 413
- “Improvements and general issues for ContactManager 7.0.6” on page 413
- “Known issues and limitations for ContactManager 7.0.6” on page 414

Release Information for ContactManager 7.0.6

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you are upgrading and you skipped one or more releases of ContactManager, be sure to read the release notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 7.0.6” on page 412.

This release of Guidewire ContactManager is 7.0.6.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 6.0.7 or later.

Installing ContactManager 7.0.6

For general installation information, see “Installing ContactManager” in the ClaimCenter Contact Management Guide.

For versions of ContactManager prior to 7.0.6 that you have skipped, see:

- “Guidewire ContactManager 7.0.5 release notes” on page 416
- “Guidewire ContactManager 7.0.4 release notes” on page 423
- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include sections for ContactCenter release notes. For example, see the release note archive in the *New and Changed Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Issues and major changes for ContactManager 7.0.6

This topic contains issues and major changes that might affect your installation.

- “Base PCF file changes for ContactManager 7.0.6” on page 413
- “Changes in this release provided in Upgrade Diff report” on page 311

Base PCF file changes for ContactManager 7.0.6

All links below require that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

ContactManager release 7.0.5 to 7.0.6

- To view a report of the changes in the base PCF files in the `modules/ab` directory, refer to the original ContactManager 7.06 release notes.
- To view a report of the changes in the base PCF files in the `modules/pl` directory, refer to the original ContactManager 7.06 release note.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and general issues for ContactManager 7.0.6

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ID	Description
Contact App	
CTC-2032	Previously, editing a contact's <code>PrimaryAddress</code> did not by itself cause the <code>ABContactChanged</code> event to be fired. This problem has been fixed by adding all properties in the <code>Address</code> entity to the history rules for primary and secondary addresses.
CTC-2065	Adding additional criteria to the <code>Address</code> entity in <code>search-config.xml</code> now works correctly.
Core	
CTC-2509	There was a problem with name related constraints for <code>ABPerson</code> entities not being enforced. If an <code>ABPerson</code> subtype had both <code>Name</code> and <code>LastName</code> set, both fields would be saved to the database, causing a consistency check to fail. This issue has been fixed. The <code>Name</code> field is now set to <code>null</code> on commit to the database, and a warning message is logged.
Integration – BC, Integration – CC, Integration – PC	
CTC-1887	When a validation error happened in the commit of an <code>ABContact</code> , <code>ContactManager</code> was throwing a retryable exception. The client application was then retrying the message and ultimately suspending the message transport due to the repeated failures. <code>ContactManager</code> now returns a non-retryable exception in this case, enabling the client application to handle the failure properly.
Integration – CC	
CTC-1922	You can now extend the data returned for related contact searches in <code>ContactManager</code> . The classes <code>RelatedContactInfoContainer</code> and <code>ABContactAPIRelatedContact</code> now have the annotation <code>@Export</code> and can be edited.

Miscellaneous

CTC-1375	In the base configuration, the default logging directory specified in <code>logging.properties</code> now matches the logging directory setting for the configured log files.
CTC-1753	There was an issue that caused erroneous consistency check failures after merging contacts. Retired contacts were not being filtered from the consistency checks that ensure that addresses are not shared by contacts. This issue has been fixed.

Known issues and limitations for ContactManager 7.0.6

This topic describes known issues with this release of Guidewire ContactManager. It has the following topics:

- “ContactManager 7.0.6 known issues” on page 414
- “Studio/Platform known issues for ContactManager 7.0.6” on page 414

Note: For maintenance releases, Guidewire often defers fixing configuration issues if they require merging files during the upgrade. Workarounds to many of these issues are listed in the following sections. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager 7.0.6 known issues

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the `RelatedABContactSearchCriteria` entity and the `CriteriaDef` elements that are defined for it in `search-config.xml` is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not check to make sure the relationship does not already exist, and therefore can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configurations of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one `ContactContact` entity in this case.

Email field validation in ContactManager not the same as in core applications (CTC-1339)

Issue – The entry in `fieldvalidators.xml` in ContactManager is more restrictive than in the core applications. This difference can cause issues while creating a contact in a core application.

Workaround – Change the ContactManager version of this validator to match the validators in the core products. Use Guidewire Studio™ to edit the entry in `fieldvalidators.xml` for `Validator.Email`, changing its value from the default `".+@.+\\.+."` to the value in the core applications, `".+@.+"`.

Studio/Platform known issues for ContactManager 7.0.6

Issues with Internet Explorer 9

Issue – If you are using the Internet Explorer 9 browser, it is possible to see issues such as screen flickering or an incorrect tab order for fields. According to public reports, Internet Explorer 9 exhibits these and other issues with a variety of web sites and web applications.

Workaround – Because this is the behavior of the Internet Explorer 9 rendering engine, Guidewire cannot address these issues. However, there are reports of an Internet Explorer 9 workaround that reduces these issues. In Internet Explorer 9, you can try to change the use of software or hardware rendering by toggling the **Accelerated Graphics** option on the **Advanced** tab of the **Internet Options** dialog.

Studio Rules do not use correct capitalization for root object's name (PL-10740)

Issue – Rule set root objects are not named with first letter llowercased.

Workaround – Guidewire is aware of this issue.

User interface cannot handle starting multiple instances of a batch process (PL-12372)

Issue – The user interface cannot handle starting multiple instances of a batch process.

Workaround – To execute multiple instances of a batch process, start them from the command line. Also, to allow multiple instances to be run simultaneously, you must ensure that the `BatchProcess.isExclusive()` method returns `false`.

Type system refresh after PCF page title change does not update corresponding menu label (PL-13057)

Issue – The type system refresh after a PCF page title change does not update the corresponding menu label.

Workaround – After updating a page title, restart the server to refresh menu labels and avoid null pointer exceptions due to stale references.

Length limitation on entity localization table names (PL-13360)

Issue – Entity localization table names have a shorter, 16-character limit than other entity tables. If the localization table name exceeds the 16-character limit, the error message indicates incorrectly that 18 characters are allowed.

Workaround – Ensure that the localization `tableName` property specified in the entity extension file is less than 16 characters.

Cannot make a field from a delegate into a localized column (PL-13761)

Issue – You cannot make a field from a delegate into a localized column.

Workaround – Move the column to be localized off the delegate and onto each of the implementing entities. Then, to make the column appear as though it exists on the delegate, define an enhancement property on the delegate that *delegates* to the appropriate column, depending on the implementing entity.

Problem with regen-java-api command and JAR files (PL-16351)

Issue – If you run the `ContactManager/bin/gwpc regen-java-api` command, ContactManager creates a `ContactManager/java-api/lib` directory and puts JAR files for Java development in this directory. However, ContactManager does not always perform this task correctly, resulting in some of the generated JAR files not being copied to this directory.

Workaround – For missing library files, copy or import into your Java development environment the missing library files from the following directory:

```
ContactManager/admin/lib
```

Copy them into the following directory:

```
ContactManager/java-api/lib
```

Renaming method or property throws ParseResultsException (PL-16633)

Issue – If you rename a property or a method or you change a method signature, and a workflow references that property or method in a Gosu field, ContactManager throws `ParseResultsException`. This behavior is intended.

Workaround – Restart the workflow engine. To do so:

1. Log into ContactManager by using an administrative account.
2. Access **Internal Tools**→**Reload**.
3. Click **Reload Workflow Engine**.

In some languages, web browsers render column headers of list views improperly (PL-18027)

Issue – In some languages, web browsers render some column headers of list views improperly if their column widths are specified too narrowly in their PCF definitions. For example, sometimes a numeric column is specified with a variable width of 1%. This narrow setting forces the browser to render the column too narrowly for the text of the translated column heading.

Workaround – Edit the PCF file that defines the column and clear the value from the width property. Without a specified value for the column width, browsers render the column widely enough to display the full text of the translated column heading.

Database upgrade does not handle nullable to non-nullable columns with a default value for subtypes (PL-23104)

Issue – For entity definitions, the automatic database upgrade converts nullable columns to non-nullable with a default value successfully. However, this column type conversion is not possible for columns in subtype definitions. ContactManager implements non-nullable columns on subtype in the database as nullable because that column must have null values for rows that represents instances of other subtypes.

Workaround – Write a version trigger to populate the column with the default value for existing rows for the subtype. After you upgrade, ContactManager enforces the column value to be non-nullable with the default value for new rows of the subtype.

New transport plugin definitions do not show in the list of valid transport plugins (PL-23317)

Issue – Newly implemented transport plugin definitions do not show in the list of valid transport plugins displayed by clicking the light-bulb icon next to the Transport Plugin field in the messaging destination editor.

Workaround – Restart Studio.

Find in Resources fails for resources under Data Model Extensions or Web Resources (PL-23320)

Issue – In Studio, the **Find in Resources** option does not work for resources that are in Data Model Extensions or Web Resources.

Workaround – To see the full resource path, hover over the resource name in the tab of the resource editor. Then, navigate in the resource pane on the left to find the resource in the resource hierarchy.

Guidewire ContactManager 7.0.5 release notes

These release notes contain the following:

- “Release Information for ContactManager 7.0.5” on page 416
- “Installing ContactManager 7.0.5” on page 417
- “Support” on page 21
- “Issues and major changes for ContactManager 7.0.5” on page 417
- “Improvements and general issues for ContactManager 7.0.5” on page 418
- “Known issues and limitations for ContactManager 7.0.5” on page 420

Release Information for ContactManager 7.0.5

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you are upgrading and you skipped one or more releases of ContactManager, be sure to read the release notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 7.0.5” on page 417.

This release of Guidewire ContactManager is 7.0.5.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 6.0.7 or later.

Installing ContactManager 7.0.5

For general installation information, see “Installing ContactManager” in the ClaimCenter Contact Management Guide.

For versions of ContactManager prior to 7.0.5 that you have skipped, see:

- “Guidewire ContactManager 7.0.4 release notes” on page 423
- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include sections for ContactCenter release notes. For example, see the release note archive in the *New and Changed Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Issues and major changes for ContactManager 7.0.5

This topic contains issues and major changes that might affect your installation.

- “Base PCF file changes for ContactManager 7.0.5” on page 417
- “Rules changes for ContactManager 7.0.5” on page 418
- “Changes in this release provided in Upgrade Diff report” on page 311

Base PCF file changes for ContactManager 7.0.5

All links below require that the `ReleaseNotes_files` directory be on your local disk in the same directory as this release notes file.

ContactManager release 7.0.4 to 7.0.5

- To view a report of the changes in the base PCF files in the `modules/ab` directory, refer to the original ContactManager 7.0.5 release notes.
- To view a report of the changes in the base PCF files in the `modules/p1` directory, refer to the original ContactManager 7.0.5 release notes.

Rules changes for ContactManager 7.0.5

ContactManager release 7.0.4 to 7.0.5

- There are no changes to the base rules in the `modules/ab` directory.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the [Guidewire Community](#).

See also

- “Support” on page 21

Improvements and general issues for ContactManager 7.0.5

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ID	Description
Contact App	
CTC-1747	The <code>LinkableExtensionUpdateLinkIdVersionTrigger</code> upgrade trigger sets LinkID values on all entities that implement <code>ABLinkable</code> before the upgrade process runs. However, the trigger was unable to create a LinkID for tables that did not have a LinkID column. The trigger can now create the LinkID column in a table if the column does not already exist.
Command line tools	
PL-20378	With this release, the <code>verify-types</code> command-line tool does not stop verification if a failure, such as a programming error, occurs while verifying a specific type. The tool reports all errors, warnings and failures.
Database support	
PL-21842	With this release, the <code>system_tools</code> command has additional options for submitting batch jobs that report database performance.
PL-23523	For Oracle, you can configure new LOB columns to use SecureFile LOBs or compressed SecureFile LOBs instead of the default BasicFile LOBs. You can configure the LOB type for the entire database or for specific tables only. For more information, see “Configuring Oracle LOB Types” in the <i>ClaimCenter Installation Guide</i> .
Database upgrade	
PL-23504	<p>With this release, a database upgrade in a development environment records checkpoints of upgrade triggers that complete successfully. You can restart a failed database upgrade, and it resumes with the upgrade trigger that failed. The restart feature helps you test your upgrade with realistically large data sets. You avoid time spent to restore the database and time spent to run upgrade triggers that work successfully.</p> <p>To restart a test database upgrade from a checkpoint reached in an earlier upgrade, roll back manually any database changes that occurred during the upgrade trigger that failed. In addition, verify that you resolved the problem that caused the trigger to fail before restarting. A test run of your upgrade is successful only when it runs from start to finish without a restart. Never use the restart feature of database upgrade in a production environment.</p>
PL-23686	<p>Generating table statistics during upgrade is now optional for Oracle databases. This change does not affect statistics generation on the Microsoft SQL Server and H2 development databases.</p> <p>You set this new option in the <code>config.xml</code> file. In the <code><database></code> block, there is an <code><upgrade></code> block that contains configuration information for the overall database upgrade. There is a new attribute in the <code><upgrade></code> block, <code>updatestatistics</code> with default value <code>true</code>. If you set this attribute to <code>false</code>, statistics will not be updated for that database during upgrade.</p> <p>If you do not update statistics during upgrade, you are then responsible for updating statistics by using the command-line batch process. Most of the time, the incremental statistics update is sufficient. For example:</p>

`maintenance_tools -password password -startprocess incrementaldbstats`

If statistics are not updated during the upgrade, you see a warning that recommends that you run the database statistics batch process in incremental mode. Additionally, the UpgradeInfo page shows that statistics were not updated as part of the upgrade. This page also reports the runs of the statistics batch process, and incremental runs are shown.

The UpgradeInfo page does not identify the following case: You ran another upgrade with `updatestatistics=true` since running a previous update with `updatestatistics=false`, but you did not update statistics first.

When you click the Download button on the UpgradeInfo page, you get a more detailed UpgradeInfo report.

- The UpgradeInfo report shows the value of the `updatestatistics` attribute at the time of upgrade.
- Additionally, the UpgradeInfo report shows the update statistics SQL statements that were skipped as part of the upgrade. Normally, you do not need to consult this list because running the database statistics batch process in full mode will cover these statements.

Note: The incremental mode might not capture all cases that would have been run during an upgrade with `updatestatistics=true`.

Entities/Metadata

PL-24640 Resolved an issue with the Gosu type system that led to an infinite loop and a stack overflow exception when constructing an entity type. The issue occurs in earlier releases if an entity uses `<implementsInterface>` and that interface has a method that returns an array of the same entity.

Gosu

PL-24177 Resolved an issue that caused the `exists` method to throw an exception when the where clause included a Guidewire boxed Boolean instead of the Java primitive type `boolean`.

PL-24095, PL-24478 Fixed an issue in which casting errors were thrown when Gosu code called a Java method that expected a specific entity, such as Document, as a parameter. The system would throw an error similar to the following:
`gw.lang.parser.EvaluationException:
 com.guidewire.commons.metadata.proxy._generated.impl.Document cannot be cast to
 com.guidewire.pc.external.entity.Document`
 This error was caused by the way in which Guidewire Java code was making a reflective method call to Gosu and then handling external entity types. Those reflective method calls no longer require casting of the external entity types.

Integration

PL-23132 There was a problem with starting a listening thread on JMS when the application server was the latest version of WebSphere. This problem has been fixed for WebSphere 7 and WebLogic 10.3.5 and later. Now you can use a listening thread with a startable plugin on these versions of WebSphere and WebLogic.

Integration – BC, Integration – CC, Integration – PC

CTC-1700 Updated the `ClientSystemPlugin` plugin implementations in `ContactManager` to handle unexpected exceptions. Rather than suspend the message transport, `ContactManager` now logs the exception and continues running as usual. The affected plugin implementations are `BCBillingSystemPlugin`, `CCClaimSystemPlugin`, and `PCPolicySystemPlugin`.

Localization

PL-24462 In previous releases, `ContactManager` incorrectly handled Imperial dates that equate to Gregorian Dates in the year 1950, as well as other ranges of old dates. The issue occurred because American military bases used day-light saving time in Japan beginning in 1948, which differed from timekeeping used by the general public. With this release, `ContactManager` handles old Imperial dates correctly.

Management plugin

PL-16543 With this release, the counters `NumActiveDBConnections` and `NumIdleDBConnectionsmanagement` are integers instead of strings, so management tools that connect by using JMX can plot them now on graphs.

Performance

CTC-1828	The ContactManager Message.eix file contains indexes to speed up the messaging subsystem when it searches for messages with a particular type of primary object, such as ABContact. Performance testing found that the existing indexes could be improved. Therefore, improved indexes have been added to Message.eix, resulting in faster query plans and message processing.
Persistence	
PL-23941	For workflow-related distributed work queues, the logging level of the message “WDW processing workitem: ...” changed from INFO to DEBUG to reduce noise in the log files.
Web - UI/Runtime	
PL-23848	PCF input elements now render currency amounts so the scale and appscale parameters are honored. For example, if appscale == 0 and the amount is the integer 1, the input element renders the amount as “1”. If appscale == 2 and the amount is the integer 1, the input element renders the amount as “1.00”.
Web services – WS-I	
PL-23514	<p>This release introduces a new method, preExecute, for use in the invocation handlers that you write for WS-I web services. In earlier releases, if you wrote an invocation handler, you unavoidably bypassed some important WS-I features. For example, the application:</p> <ul style="list-style-type: none"> • Did not enable profiling for method calls. • Did not check run level annotations, even at the class level. • Did not check web service permission annotations, even at the class level. • Did not check for duplicate external transaction IDs if those SOAP headers were present. <p>Now, you can call the preExecute method in your invocation handler to assure the application takes the preceding actions, as needed. See the <i>Integration Guide</i> for more information about invocation handlers and how to use the preExecute method.</p>
PL-23721	To detect duplicate operations from external systems, add the new annotation @WsiCheckDuplicateExternalTransaction to your WS-I web service implementation class. See the <i>Integration Guide</i> for details for using the annotation and the SOAP header for the transaction ID.

Known issues and limitations for ContactManager 7.0.5

This topic describes known issues with this release of Guidewire ContactManager. It has the following topics:

- “ContactManager known issues for release 7.0.5” on page 420
- “Studio/Platform known issues for ContactManager 7.0.5” on page 421

Note: For maintenance releases, Guidewire often defers fixing configuration issues if they require merging files during the upgrade. Workarounds to many of these issues are listed in the following sections. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager known issues for release 7.0.5

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the RelatedABContactSearchCriteria entity and the CriteriaDef elements that are defined for it in search-config.xml is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one ContactContact entity. ContactManager does not check that the relationship does not already exist. Therefore, ContactManager can create multiple, redundant ContactContact entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configurations of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one ContactContact entity in this case.

Email field validation in ContactManager not the same as in core applications (CTC-1339)

Issue – The entry in `fieldvalidators.xml` in ContactManager is more restrictive than in the core applications. This difference can cause issues while creating a contact in a core application.

Workaround – Change the ContactManager version of this validator to match the validators in the core products. Use Guidewire Studio™ to edit the entry in `fieldvalidators.xml` for `Validator.Email`, changing its value from the default `".+@.\..+."` to the value in the core applications, `".+@.+."`.

Studio/Platform known issues for ContactManager 7.0.5

Issues with Internet Explorer 9

Issue – If you are using the Internet Explorer 9 browser, it is possible to see issues such as screen flickering or an incorrect tab order for fields. According to public reports, Internet Explorer 9 exhibits these and other issues with a variety of web sites and web applications.

Workaround – Because this is the behavior of the Internet Explorer 9 rendering engine, Guidewire cannot address these issues. However, there are reports of an Internet Explorer 9 workaround that reduces these issues. In Internet Explorer 9, you can change the use of software or hardware rendering by toggling the **Accelerated Graphics** option on the **Advanced** tab of the **Internet Options** dialog.

Studio Rules do not use correct capitalization for root object's name (PL-10740)

Issue – Rule set root objects are not named with first letter lowercased.

Workaround – Guidewire is aware of this issue.

User interface cannot handle starting multiple instances of a batch process (PL-12372)

Issue – The user interface cannot handle starting multiple instances of a batch process.

Workaround – To execute multiple instances of a batch process, start them from the command line. Also, to allow multiple instances to be run simultaneously, you must ensure that the `BatchProcess.isExclusive()` method returns `false`.

Type system refresh after PCF page title change does not update corresponding menu label (PL-13057)

Issue – The type system refresh after a PCF page title change does not update the corresponding menu label.

Workaround – After updating a page title, restart the server to refresh menu labels and avoid null pointer exceptions due to stale references.

Length limitation on entity localization table names (PL-13360)

Issue – Entity localization table names have a shorter, 16-character limit than other entity tables. If the localization table name exceeds the 16-character limit, the error message indicates incorrectly that 18 characters are allowed.

Workaround – Ensure that the `tableName` property specified in the entity extension file is less than 16 characters.

US-Locations.txt file with the US geodata from Great Data has special characters that cause validation problems with United States Postal Service (USPS) data (PL-13384)

Issue – The `US-Locations.txt` file contains information that does not conform to United States Postal Service (USPS) standards for bulk mailings.

Workaround – The provided `US-Locations.txt` file is intended only for use in geocoding to identify addresses for a location. You can edit the `US-Locations.txt` file to conform to your particular address standards, and then import that version of the file instead.

GX model generated XSD cannot be parsed by JAXB (PL-13598)

Issue – XSD generated by the GX model cannot be parsed by JAXB.

Workaround – Add JAXB annotation elements to the XSD to specify the necessary metadata, such as class names, to enable JAXB to generate the Java class files. Contact Guidewire Support for a sample XSD file that is annotated this way.

Cannot make a field from a delegate into a localized column (PL-13761)

Issue – You cannot make a field from a delegate into a localized column.

Workaround – Move the column to be localized off the delegate and onto each of the implementing entities. Then, to make the column appear to be on the delegate, define an enhancement property on the delegate that *delegates* to the appropriate column, depending on the implementing entity.

Problem with regen-java-api command and JAR files (PL-16351)

Issue – If you run the `ContactManager/bin/gwpc regen-java-api` command, ContactManager creates a `ContactManager/java-api/lib` directory and puts JAR files for Java development in this directory. However, ContactManager does not always perform this task correctly, resulting in some of the generated JAR files not being copied to this directory.

Workaround – For missing library files, copy or import into your Java development environment the missing library files from the following directory:

```
ContactManager/admin/lib
```

Copy them into the following directory:

```
ContactManager/java-api/lib
```

Renaming method or property throws ParseException (PL-16633)

Issue – If you rename a property or a method or change a method signature, and a workflow references that property or method in a Gosu field, ContactManager throws `ParseException`. This behavior is intended.

Workaround – Restart the workflow engine. To do so:

1. Log into ContactManager by using an administrative account.
2. Access **Internal Tools**→**Reload**.
3. Click **Reload Workflow Engine**.

In some languages, web browsers render column headers of list views improperly (PL-18027)

Issue – In some languages, web browsers render some column headers of list views improperly if their column widths are specified too narrowly in their PCF definitions. For example, sometimes a numeric column is specified with a variable width of 1%. This narrow setting forces the browser to render the column too narrowly for the text of the translated column heading.

Workaround – Edit the PCF file that defines the column and clear the value from the width property. Without a specified value for the column width, browsers render the column widely enough to display the full text of the translated column heading.

Database upgrade does not handle nullable to non-nullable columns with a default value for subtypes (PL-23104)

Issue – For entity definitions, the automatic database upgrade converts nullable columns to non-nullable with a default value successfully. However, this column type conversion is not possible for columns in subtype definitions.

ContactManager implements non-nullable columns on subtype in the database as nullable because that column must have null values for rows that represents instances of other subtypes.

Workaround – Write a version trigger to populate the column with the default value for existing rows for the subtype. After you upgrade, ContactManager enforces the column value to be non-nullable with the default value for new rows of the subtype.

New transport plugin definitions do not show in the list of valid transport plugins (PL-23317)

Issue – Newly implemented transport plugin definitions do not show in the list of valid transport plugins. You display this list by clicking the light-bulb icon next to the Transport Plugin field in the messaging destination editor.

Workaround – Restart Studio.

Find in Resources fails for resources under Data Model Extensions or Web Resources (PL-23320)

Issue – In Studio, the **Find in Resources** option does not work for resources that are in Data Model Extensions or Web Resources.

Workaround – To see the full resource path, hover over the resource name in the tab of the resource editor. Then, navigate in the resource pane on the left to find the resource in the resource hierarchy.

Guidewire ContactManager 7.0.4 release notes

These release notes contain the following:

- “Release Information for ContactManager 7.0.4” on page 423
- “Installing ContactManager 7.0.4” on page 423
- “Support” on page 21
- “Issues and major changes for ContactManager 7.0.4” on page 424
- “Improvements and general issues for ContactManager 7.0.4” on page 424
- “Known issues and limitations for ContactManager 7.0.4” on page 426

Release Information for ContactManager 7.0.4

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you are upgrading and you skipped one or more releases of ContactManager, be sure to read the release notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 7.0.4” on page 423.

This release of Guidewire ContactManager is 7.0.4.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 6.0.7 or later.

Installing ContactManager 7.0.4

For general installation information, see “Installing ContactManager” in the *ClaimCenter Contact Management Guide*.

For versions of ContactManager prior to 7.0.4 that you have skipped, see:

- “Guidewire ContactManager 7.0.3 release notes” on page 429
- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include sections for ContactCenter release notes. For example, see the release note archive in the *New and Changed Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Issues and major changes for ContactManager 7.0.4

This topic contains issues and major changes that might affect your installation.

- “Base PCF file changes for ContactManager 7.0.4” on page 424
- “Rules changes for ContactManager 7.0.4” on page 424
- “Changes in this release provided in Upgrade Diff report” on page 311

Base PCF file changes for ContactManager 7.0.4

All links below require the `ReleaseNotes_files` directory on your local disk, in the same directory as this release notes file.

ContactManager release 7.0.3 to 7.0.4

- To view a report of the changes in the base PCF files in the `modules/ab` directory, refer to the original ContactManager 7.0.4 release notes.
- To view a report of the changes in the base PCF files in the `modules/pl` directory, refer to the original ContactManager 7.0.4 release notes.

Rules changes for ContactManager 7.0.4

ContactManager release 7.0.3 to 7.0.4

- There are no changes to the base rules in the `modules/ab` directory.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and general issues for ContactManager 7.0.4

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ID	Description
Authentication	
PL-21737	The Gosu implementation of the AuthenticationSourceCreator plugin now supports JBoss.

Build Infrastructure

PL-15509 In previous versions, Guidewire supported running build scripts only through the `gwab` command on Windows. With this version you, can run the EAR and WAR build scripts on Unix by invoking Ant directly with the following command:

```
ant -f ContactManager/modules/ant/build.xml buildScript
```

Substitute one of the following values for *buildScript*:

- `build-jboss-war` – Builds a generic WAR file for use with JBoss.
- `build-tomcat-war` – Builds a generic WAR file for use with Tomcat.
- `build-weblogic-ear` – Builds an EAR file for use with WebLogic.
- `build-websphere-ear` – Builds an EAR file for use with WebSphere.

Although you can run many build scripts by using the `gwab` command on Windows, if you invoke Ant on Unix, Guidewire supports only the build scripts in the previous list.

Command line tools

PL-20110 The command-line utility `gwab regen-soap-api` no longer fails on Japanese Windows.

Contact App

CTC-1646 Fixed an issue in which the default value of the `DuplicateContactWideSearch` configuration parameter was not set.

Core

CTC-1336 Fixed an issue in which the `DefinitiveMatchSet` definitions were causing erroneous consistency and integrity checks to be run. The deprecated `IContactAPI` enforces these checks, but the base configuration of `ContactManager` does not use `IContactAPI`. The fix was to edit `definitive-match-config.xml` and comment out these `DefinitiveMatchSet` settings.

If you are using `IContactAPI`, you can re-enable these checks by editing `definitive-match-config.xml` and removing the comments on the `DefinitiveMatchSet` settings that are commented out.

Database

PL-21554 In previous versions on SQL Server, tuning queries was difficult because DBAs could not determine what part of the application generated the queries. Now, you can enable the new `IdentifyQueriesViaComments` parameter in the `config.xml` file to provide comments with contextual information in certain SQL `Select` statements sent to the relational database.

The SQL comments are in the format:

```
ApplicationName:ActionName
```

ApplicationName is `ContactManager`.

ActionName is the name of the PCF file that submitted the SQL `Select` statement.

Database Upgrade

PL-22007 In earlier versions, `ContactManager` ran the trigger `AfterUpgradeVersionTrigger` on fresh databases, which could cause problems. With this release, `ContactManager` runs this trigger only on existing databases, not on fresh databases.

PL-22478 In earlier versions, on SQL Server queries failed if they contained a large number of CASE conditions. Now, `ClaimCenter` breaks up queries with more than 125 CASE conditions into nested CASE clauses or into separate queries.

PL-22686 With this release, the validation SQL run by the upgrade version checks is now included in the downloads that you obtain from the Upgrade Info page.

PL-22934 This release fixes an issue that caused a SQL failure during database upgrade of the `InstrumentedBatchJobID` column on the `instrumentedworkertask` table.

PL-23044 With this release, the database upgrader now supports table and index partitioning. Specify which tables and indexes you want to have partitioned, and the upgrader creates any new table and new indexes appropriately. Existing tables and indexes remain unchanged, regardless of the discrepancy between the metadata configuration file and the database schema.

Entities/Metadata

PL-22866 Calling an API method that attempts to modify a read-only bundle now shows an immediate error.

PL-22597	Previously, if you defined an entity type that implemented a Gosu interface, problems occurred if the interface had a method that returned an array of that entity type. None of the interface features were accessible on the entity that implemented the interface, and class cast exceptions could occur. Now, the interface features are accessible and class cast exceptions do not occur.
Gosu	
PL-22469	When an entity contains an <code>implementsInterface</code> element, that entity type is supposed to implement the interface specified in the <code>implementsInterface</code> element. Previously, methods inherited by the interface could be invoked, but references to the entity could not be assigned to variables of the interface type. This issue fixes the problem so that the entity type truly implements the interface, preserving assignability.
Integration - BC, Integration - CC, Integration - PC	
CTC-1307	The find duplicates code run by the batch process was configurable to use a narrow or wide search in the <code>DuplicateContactsWideSearch</code> configuration parameter. This configuration parameter is now used by the <code>ABContactAPI findDuplicates</code> method as well, so the behavior of the API call and the batch process is now the same.
Localization	
CTC-1429	Updated all messaging destinations to use display keys for the Name field to support translation into different languages.
Miscellaneous	
CTC-1384	The Merge Contacts permission to handle duplicate contacts was added to the Super User role in ContactManager 7.0.0. At the time, no upgrade trigger was added to upgrade existing installations. There is now an upgrade trigger to add the Merge Contacts permission to the Super User role. This trigger has no effect if you have already added this permission to the Super User role. This permission has not been added to any other role.
Web - Other	
PL-18559	Added Alt text, alternative tooltip text, for the following: <ul style="list-style-type: none"> • <code>BooleanRadioInput</code> • <code>PickerLink</code> • <code>DocumentationCell</code>

Known issues and limitations for ContactManager 7.0.4

This topic describes known issues with this release of Guidewire ContactManager. It has the following topics:

- “ContactManager known issues for release 7.0.4” on page 426
- “Studio/Platform Known Issues for ContactManager 7.0.4” on page 427

Note: For maintenance releases, Guidewire often defers fixing configuration issues if they require merging files during the upgrade. Workarounds to many of these issues are listed in the following sections. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager known issues for release 7.0.4

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the `RelatedABContactSearchCriteria` entity and the `CriteriaDef` elements that are defined for it in `search-config.xml` is not supported in this version of ContactManager.

Workaround – Do not use them. Guidewire is aware of this issue.

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not check to make sure the relationship does not already exist. Therefore, ContactManager can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configurations of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one Contact entity in this case.

Email field validation in ContactManager not the same as in core applications (CTC-1339)

Issue – The entry in `fieldvalidators.xml` in ContactManager is more restrictive than in the core applications. This difference can cause issues while creating a contact in a core application.

Workaround – Change the ContactManager version of this validator to match the validators in the core products. Use Guidewire Studio™ to edit the entry in `fieldvalidators.xml` for `Validator.Email`, changing its value from the default `".+@.\.+.+"` to the value in the core applications, `".+@.+"`.

Studio/Platform Known Issues for ContactManager 7.0.4

Issues with Internet Explorer 9

Issue – If you are using the Internet Explorer 9 browser, it is possible to see issues such as screen flickering or an incorrect tab order for fields. According to public reports, Internet Explorer 9 exhibits these and other issues with a variety of web sites and web applications.

Workaround – Because this is the behavior of the Internet Explorer 9 rendering engine, Guidewire cannot address these issues. However, there are reports of an Internet Explorer 9 workaround that reduces these issues. In Internet Explorer 9, you can try to change the use of software or hardware rendering. To do so, toggle the **Accelerated Graphics** option on the **Advanced** tab of the **Internet Options** dialog.

Studio Rules do not use correct capitalization for root object's name (PL-10740)

Issue – Rule set root objects are not named with first letter lowercased.

Workaround – Guidewire is aware of this issue.

User interface cannot handle starting multiple instances of a batch process (PL-12372)

Issue – The user interface cannot handle starting multiple instances of a batch process.

Workaround – To execute multiple instances of a batch process, start them from the command line. Also, to allow multiple instances to be run simultaneously, you must ensure that the `BatchProcess.isExclusive()` method returns `false`.

Type system refresh after PCF page title change does not update corresponding menu label (PL-13057)

Issue – The type system refresh after a PCF page title change does not update the corresponding menu label.

Workaround – After updating a page title, restart the server to refresh menu labels and avoid null pointer exceptions due to stale references.

Length limitation on entity localization table names (PL-13360)

Issue – Entity localization table names have a shorter, 16-character limit than other entity tables. If the localization table name exceeds the 16-character limit, the error message indicates incorrectly that 18 characters are allowed.

Workaround – Ensure that the localization `tableName` property specified in the entity extension file is less than 16 characters.

US-Locations.txt file with the US geodata from Great Data has special characters that cause validation problems with United States Postal Service (USPS) data (PL-13384)

Issue – The US-Locations.txt file contains information that does not conform to United States Postal Service (USPS) standards for bulk mailings.

Workaround – The provided US-Locations.txt file is intended only for use in geocoding to identify addresses for a location. You can edit the US-Locations.txt file to conform to your particular address standards, and then import that version of the file instead.

GX model generated XSD cannot be parsed by JAXB (PL-13598)

Issue – XSD generated by the GX model cannot be parsed by JAXB.

Workaround – Add JAXB annotation elements to the XSD to specify the necessary metadata, such as class names, to enable JAXB to generate the Java class files. Contact Guidewire Support for a sample XSD file that is annotated this way.

Cannot make a field from a delegate into a localized column (PL-13761)

Issue – You cannot make a field from a delegate into a localized column.

Workaround – Move the column to be localized off the delegate and onto each of the implementing entities. Then, make the column appear as though it exists on the delegate. To do so, define an enhancement property on the delegate that delegates to the appropriate column, depending on the implementing entity.

Problem with regen-java-api command and JAR files (PL-16351)

Issue – If you run the ContactManager/bin/gwpc regen-java-api command, ContactManager creates a ContactManager/java-api/lib directory and puts JAR files for Java development in this directory. However, ContactManager does not always perform this task correctly, resulting in some of the generated JAR files not being copied to this directory.

Workaround – For missing library files, copy or import into your Java development environment the missing library files from the following directory:

```
ContactManager/admin/lib
```

Copy them into the following directory:

```
ContactManager/java-api/lib
```

Renaming method or property throws ParseException (PL-16633)

Issue – If you rename a property or a method or change a method signature, and a workflow references that property or method in a Gosu field, ContactManager throws ParseException. This behavior is intended.

Workaround – Restart the workflow engine. To do so:

1. Log into ContactManager using an administrative account.
2. Access **Internal Tools**→**Reload**.
3. Click **Reload Workflow Engine**.

Guidewire ContactManager 7.0.3 release notes

These release notes contain the following:

- “Release Information for ContactManager 7.0.3” on page 429
- “Installing ContactManager 7.0.3” on page 429
- “Support” on page 21
- “Issues and major changes for ContactManager 7.0.3” on page 429
- “Improvements and general issues for ContactManager 7.0.3” on page 430
- “Known issues and limitations for ContactManager 7.0.3” on page 433

Release Information for ContactManager 7.0.3

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you are upgrading and you skipped one or more releases of ContactManager, be sure to read the release notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 7.0.3” on page 429.

This release of Guidewire ContactManager is 7.0.3.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 6.0.7 or later.

Installing ContactManager 7.0.3

For general installation information, see “Installing ContactManager” in the *ClaimCenter Contact Management Guide*.

For versions of ContactManager prior to 7.0.3 that you have skipped, see:

- “Guidewire ContactManager 7.0.2 release notes” on page 436
- “Guidewire ContactManager 7.0.1 release notes” on page 447

For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include sections for ContactCenter release notes. For example, see the release note archive in the *New and Changed Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Issues and major changes for ContactManager 7.0.3

This topic contains issues and major changes that might affect your installation.

- “Base PCF file changes for ContactManager 7.0.3” on page 430
- “Rules changes for ContactManager 7.0.3” on page 430
- “Changes in this release provided in Upgrade Diff report” on page 311

Base PCF file changes for ContactManager 7.0.3

All links below require the `ReleaseNotes_files` directory for release 7.0.3 on your local disk.

ContactManager release 7.0.2 to 7.0.3

- To view a report of the changes in the base PCF files in the `modules/ab` directory, refer to the original ContactManager 7.0.3 release notes.
- To view a report of the changes in the base PCF files in the `modules/pl` directory, refer to the original ContactManager 7.0.3 release notes.

Rules changes for ContactManager 7.0.3

ContactManager release 7.0.2 to 7.0.3

- There are no changes to the base rules in the `modules/ab` directory.

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and general issues for ContactManager 7.0.3

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ID	Description
Archiving	
PL-19035	Added a Download button to the (Server Tools) Archive Info page to more easily enable the collection and sending of archiving error information.
PL-20041	Fixed an issue in which the archiving schema (archiving.XSD) did not match the archiving output XML.
PL-20690	A null pointer exception sometimes occurred during restore of archived claims. A more informative error message is now shown.
PL-21380	The <code>regen-xsd</code> command generated an XSD file with errors. The command has been updated to produce a valid XSD file.
Batch Processes	
PL-20755	Fixed an issue in which worker threads could unexpectedly quit if any database or JDBC error occurred during a time span longer than the maximum wait period.
Command Line Tools	
PL-19768	Fixed an issue that occurred while attempting to execute the <code>verify-types</code> command on a 64-bit JVM, which caused an <code>OutOfMemoryError</code> .
Consistency Checker	
PL-16205	Requesting a run of the database consistency checks by using the <code><database></code> element attribute <code>checker="true"</code> in the <code>config.xml</code> file is a deprecated feature. If you do so, ContactManager now prints a warning message in the log warning that this feature is deprecated. The message also instructs you to use the Info Page or command line to run the consistency checks as a batch job.

Contact Domain

- CTC-1138 The SetVendorsPrimaryAddressBatchGeocode rule was incorrectly resetting an ABContact object's status to BatchGeocode if anything in the object changed, even if the change did not affect geocoding. This rule now does not reset an ABContact object's status to BatchGeocode unless a change to the object affects geocoding.

Core

- CTC-618 Removed print statements that were producing unnecessary clutter in the console log messages when starting ContactManager.

Database

- PL-8729 Guidewire removed the configuration parameter TableEstimatePercent from the config.xml file. Instead, use the samplingpercentage attribute of the database statistics element in config.xml, as the following sample XML code shows.
- ```
<databasestatistics samplingpercentage="20">
```
- PL-10469 With this release, Guidewire applications do not start if the CurrentEncryptionPlugin parameter in config.xml specifies an encryption plugin implementation that does not exist.
- PL-10468 Fixed an issue on the **Database Storage Information** page regarding the table display whenever switching from **Index Physical Statistics** to **Tables and Indexes**.
- PL-14490 The data distribution tables now include a column for the internal major version number of the database schema.
- PL-17742 On Oracle, changed database upgrade logic by providing additional optimizer hints to help improve upgrade performance.
- PL-17835 Behavior has changed with evicting a database connection from the connection pool when an exception occurs. Guidewire applications now determine whether an exception was fatal to a connection before marking it for eviction. In addition, whenever the application marks a connection for eviction, the application logs the reason for the eviction. Constraint violations no longer cause evictions.
- PL-20546 In data distribution tables, the size of numeric columns increased to hold larger values.
- PL-20584 On Oracle, the Guidewire Profiler changed to help optimize the capture of the range of AWR snapshots.
- PL-21145 Replaced the String value on DBNullConstraint Exception with display key Java.Database.DBException.NullConstraintViolation to enable localization of the message.
- PL-21448 On SQL Server, ContactManager now provides additional information on database connections, which can then be captured and analyzed in the Server DMV Snapshot screen.

**Database Support**

- PL-19689 Guidewire has changed the mechanism for creating the database performance reports for Oracle and SQL Server. You no longer have to click a button and wait for the report to be generated and packaged in a ZIP file for immediate download. The report is now generated by a background batch process that stores the completed report in a new table named ab\_dbperfreport. This change requires a database upgrade to create this new table.

**Database Upgrade**

- PL-16977 Previously after a database upgrade, ContactManager logged differences between the metadata configuration and the upgraded database schema at the INFO logging level. Now, ContactManager logs differences at the WARN logging level.
- PL-20200 Guidewire has implemented optimization on the database upgrade. This optimization improves the performance of new database creation, as well as database changes during minor and major upgrades.
- PL-21310 Fixed an issue that caused database upgrades to fail if the WorkFlowWorkItem table was populated. Upgrade succeeded if the table was empty. The fix involved changing the WorkFlowWorkItem entity attribute from Final to Extendable.
- PL-21357 The **Upgrade Info** page and download now provide more consistent information about the steps that are being performed during an automated database upgrade.
- PL-21387 Corrected an issue that caused a null-pointer exception (NPE) in version triggers during a database upgrade.



## Email

- PL-13582 A problem prevented files with Unicode file names attached to email from reaching email clients. Now, email clients that handle attachments with Unicode file names, such as Microsoft Outlook, receive the attachments from ContactManager and send them correctly. If your email client does not handle Unicode file names, you must modify the `EmailMessageTransport` plugin to convert attachment file names to use Latin characters only.

## Entities/Metadata

- PL-20127 Guidewire has corrected an issue with setting the `appscale` attribute for the Money data type. The `appscale` attribute worked for `currencyamount` columns, but not for money columns. The setting, specified in `datatypes.xml`, was being ignored when a column of type money was retrieved. This setting is now working. The `appscale` attribute controls the number of digits shown to the right of the decimal point for money or currency in single currency mode. It is similar to the `scale` attribute. The `appscale` attribute must be smaller than the `scale` setting. If defined, the `appscale` attribute overrides the `scale` attribute.
- PL-18068 Fixed a problem with verification of `typelist` typecodes on server startup. `Typelist` typecodes that match regardless of case fail validation. For example, the following two typecodes would fail validation:
- `MyCode` and `Mycode`
- PL-20193 When an element of an owned array changes, the parent changes as well. For `effdated` owned arrays, the wrong parent changed. Now, the correct parent changes. This issue also affects `effdated` owned one-to-ones, because they are implemented as owned arrays.

## File Export

- PL-21546 Exported CSV files no longer contain UTF-8 BOM (byte order mark) characters at the beginnings of files that do not use UTF-8 encoding.

## Global Cache

- PL-20332 A new configuration parameter, `GlobalCacheDetailedStats`, determines whether to collect detailed statistics for the global cache. Detailed statistics are data that ContactManager collects to explain why items are evicted from the cache. Basic statistics, such as miss ratio, are still collected regardless of the value of this parameter. The `GlobalCacheDetailedStats` parameter is set to `false` by default. Set the parameter to `true` to help tune your cache. At runtime, use the **Management Beans** page to enable the collection of detailed statistics for the global cache. Whenever the `GlobalCacheDetailedStats` parameter is disabled, the **Evict Information** and **Type of Cache Misses** graphs are not visible.

## Gosu

- PL-20961 In Gosu you can now subclass an inner class that a supertype declares.

## Messaging

- PL-18715 The Studio **Messaging** editor contained hard-coded strings for destination names, which made localization of destination names shown in the **Admin** tab difficult. Now, destination names are display keys. Therefore, destination names are included in the translatable resources of the application. Display keys for destination names are in the `Java.MessageDestination` level of the display key namespace.
- PL-13887 The `IMessagingToolsAPI` has a new method to retrieve the status of a messaging destination. The method signature is:
- ```
getDestinationStatus(destID : int) : String
```

Miscellaneous

- CTC-478 Spelling errors in the display keys for Law Firm Specialty and Medical Firm Specialty have been fixed.

Plugins

- PL-21409 In earlier versions, the User Authentication Service plugin `AuthenticationServicePlugin` threw four exceptions that `LoginForm.java` caught and showed. There was no standard mechanism to add custom exceptions to the plugin and the login form. Now, you can modify the User Authentication Service plugin and throw `DisplayableLoginException`.

Staging Tables

| | |
|--------------------------------|---|
| PL-20303 | ContactManager provides two new consistency checks: “One-to-one non-null check” and “Edge foreign key non-null check”. |
| Studio IDE | |
| PL-19308 | An issue occurred with Studio because some source control systems require a file or directory in a directory controlled by Studio. Studio interfered with the special, source control files. Now, Studio safely ignores third party files and directories required by some source control systems. |
| PL-20531 | Fixed a problem that could cause Studio to not properly render selected text and caret positions in files containing true tab characters. |
| PL-21216 | Studio can generate a new warning message during PCF verification, (Verify All). It generates the warning if the number of attempted verification passes exceeds the number set for Section Inclusion Limit in Tools→Verification Options→PCF Verification Options . |
| PL-21296 | Modified the Studio PCF Editor to make the PCF Toolbox show subcategories for all Application customized widgets that you can use in widget filtering. |
| PL-21447 | Corrected an issue in which the Studio Gosu editor periodically froze for a brief period of time on large files if you typed quickly. |
| Web – UI/Runtime | |
| PL-20150 | Modified ContactManager event-processing logic to skip events for removed widgets. For example, if the values of two widgets have been changed in the same form submission, there will be two onChange events in the event queue. If the handler of the first event implicitly removes the widget that is the source of the second event, the second event will no longer be processed. The result is that the associated onChange handler will not be invoked to avoid errors. |
| Web Services | |
| PL-21158 | The Guidewire Profiler previously did not profile WS-I web services. Now whenever you select a WS-I webservice to profile, ContactManager provides profile information. |
| Web Services - WSI | |
| PL-18729 | It is now possible to invoke local web services using the wsi.local mechanism from Studio without a running server. |
| PL-20174 | It is now possible to run WSI Web Services on a server other than the batch server in a cluster. |
| XML Element - XSD types | |
| PL-20076 | Fixed an XML parsing bug that caused an xs:choice with a zero-width match to not be handled properly. |

Known issues and limitations for ContactManager 7.0.3

This section describes known issues with this release of Guidewire ContactManager.

- “ContactManager known issues for release 7.0.3” on page 433
- “Studio/Platform Known Issues for ContactManager 7.0.3” on page 434

Note: For maintenance releases, Guidewire often defers fixing configuration issues if they require merging files during the upgrade. Workarounds to many of these issues are listed in the following sections. The goal of this policy is to make upgrades as straightforward as possible.

ContactManager known issues for release 7.0.3

RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

Issue – Using the RelatedABContactSearchCriteria entity and the CriteriaDef elements that are defined for it in search-config.xml is not supported in ContactManager 7.0.3.

Workaround – Do not use them. Guidewire is aware of this issue.

Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

Issue – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one ContactContact entity. ContactManager does not check to make sure the relationship does not already exist. Therefore ContactManager can create multiple, redundant ContactContact entities for the same relationship if it receives messages changing the relationship between two contacts.

Workaround – In the base configurations of ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one ContactContact entity in this case.

Email field validation in ContactManager not the same as in core applications (CTC-1339)

Issue – The entry in fieldvalidators.xml in ContactManager is more restrictive than in the core applications. This difference can cause issues while creating a contact in a core application.

Workaround – Change the ContactManager version of this validator to match the validators in the core products. Use Studio to edit the entry in fieldvalidators.xml for Validator.Email, changing its value from the default ".+@.\.\.+" to the value in the core applications, ".+@.+".

ContactManager 7 might not find duplicate contacts when ClaimCenter 6 requests a relink (CTC-1333)

Issue – When ClaimCenter 6 is integrated with ContactManager 7, if a ClaimCenter contact becomes unlinked, the ClaimCenter user can click the Relink button to link the contact again. What happens then is that ContactManager checks first to see if there are duplicate contacts and creates a new contact only if there are no matches. However, when ContactManager receives a link request from ClaimCenter 6, ContactManager uses older matching logic for finding duplicates. This older matching logic might not find a matching contact that the newer matching logic might find after a link request from ClaimCenter 7. Therefore, ContactManager might create a new contact when it receives a link message from ClaimCenter 6 that it would not create if the link request came from ClaimCenter 7. If ContactManager received the same link request from ClaimCenter 7, it might find definitive or potential matches instead, and not create a new contact.

Workaround – Guidewire is aware of this issue.

Duplicate key warning when starting ContactManager server with empty database (CTC-1341)

Issue – When starting the ContactManager server with an empty database you might see a DBDuplicateKeyException in the logs. There is a log message preceding the exception that states, **User duplicate key exception is logged, but can be ignored.**

Workaround – As stated in the log message, you can ignore the exception.

Studio/Platform Known Issues for ContactManager 7.0.3

Issues with Internet Explorer 9

Issue – If you are using the Internet Explorer 9 browser, it is possible to see issues such as screen flickering or an incorrect tab order for fields. According to public reports, Internet Explorer 9 exhibits these and other issues with a variety of web sites and web applications.

Workaround – Because this is the behavior of the Internet Explorer 9 rendering engine, Guidewire cannot address these issues. However, there are reports of an Internet Explorer 9 workaround that reduces these issues. In Internet Explorer 9, you can try changing the use of software or hardware rendering by toggling the **Accelerated Graphics** option on the **Advanced** tab of the **Internet Options** dialog.

Studio Rules do not use correct capitalization for root object's name (PL-10740)

Issue – Rule set root objects are not named with first letter lowercased.

Workaround – Guidewire is aware of this issue.

User interface cannot handle starting multiple instances of a batch process (PL-12372)

Issue – The user interface cannot handle starting multiple instances of a batch process.

Workaround – To execute multiple instances of a batch process, start them from the command line. Also, to allow multiple instances to be run simultaneously, you must ensure that the `BatchProcess.isExclusive()` method returns `false`.

Type system refresh after PCF page title change does not update corresponding menu label (PL-13057)

Issue – The type system refresh after a PCF page title change does not update the corresponding menu label.

Workaround – After updating a page title, restart the server to refresh menu labels and avoid null pointer exceptions due to stale references.

Length limitation on entity localization table names (PL-13360)

Issue – Entity localization table names have a shorter, 16-character limit than other entity tables. If the localization table name exceeds the 16-character limit, the error message indicates incorrectly that 18 characters are allowed.

Workaround – Ensure that the `localization tableName` property specified in the entity extension file is less than 16 characters.

US-Locations.txt file with the US geodata from Great Data has special characters that cause validation problems with United States Postal Service (USPS) data (PL-13384)

Issue – The `US-Locations.txt` file contains information that does not conform to United States Postal Service (USPS) standards for bulk mailings.

Workaround – The provided `US-Locations.txt` file is intended only for use in geocoding to identify addresses for a location. You can edit the `US-Locations.txt` file to conform to your particular address standards, and then import that version of the file instead.

GX model generated XSD cannot be parsed by JAXB (PL-13598)

Issue – XSD generated by the GX model cannot be parsed by JAXB.

Workaround – Add JAXB annotation elements to the XSD to specify the necessary metadata, such as class names, to enable JAXB to generate the Java class files. Contact Guidewire Support for a sample XSD file that is annotated this way.

Cannot make a field from a delegate into a localized column (PL-13761)

Issue – You cannot make a field from a delegate into a localized column.

Workaround – Move the column to be localized off the delegate and onto each of the implementing entities. Then, define an enhancement property on the delegate that delegates to the appropriate column, depending on the implementing entity. Doing so makes the column appear as though it exists on the delegate.

Problem with regen-java-api command and JAR files (PL-16351)

Issue – If you run the `ContactManager/bin/gwpc regen-java-api` command, ContactManager creates a `ContactManager/java-api/lib` directory and puts JAR files for Java development in this directory. However, ContactManager does not always perform this task correctly, resulting in some of the generated JAR files not being copied to this directory.

Workaround – For missing library files, copy or import into your Java development environment the missing library files from the following directory:

```
ContactManager/admin/lib
```

Copy them into the following directory:

```
ContactManager/java-api/lib
```

Renaming method or property throws ParseResultsException (PL-16633)

Issue – If you rename a property or method or you change a method signature, and a workflow references that property or method in a Gosu field, ContactManager throws `ParseResultsException`. This behavior is intended.

Workaround – Restart the workflow engine. To do so:

1. Log into ContactManager using an administrative account.
2. Access **Internal Tools**→**Reload**.
3. Click **Reload Workflow Engine**.

Guidewire ContactManager 7.0.2 release notes

These release notes contain the following:

- “Release notes update: 04-June-2012” on page 436
- “Release Information for ContactManager 7.0.2” on page 436
- “Installing ContactManager 7.0.2” on page 436
- “Support” on page 21
- “Issues and major changes for ContactManager 7.0.2” on page 437
- “Improvements and general issues for ContactManager 7.0.2” on page 437
- “Known issues and limitations for ContactManager 7.0.2” on page 444

Release notes update: 04-June-2012

IMPORTANT These release notes replace the release notes that were included in the official product release. Please disregard the earlier version of the release notes.

Release Information for ContactManager 7.0.2

These release notes apply only to this release of Guidewire ContactManager.

IMPORTANT If you are upgrading and you skipped one or more releases of ContactManager, be sure to read the release notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 7.0.2” on page 436.

This release of Guidewire ContactManager is 7.0.2.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 7.0.0 or later.

Installing ContactManager 7.0.2

For general installation information, see “Installing ContactManager” on page 55.

For versions of ContactManager prior to 7.0.2 that you have skipped, see:

- “Guidewire ContactManager 7.0.1 release notes” on page 447

For versions prior to 7.0.0, see the ClaimCenter release notes for that version, which include sections for ContactCenter release notes. For example, see the release note archive in the *New and Changed Guide*.

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Issues and major changes for ContactManager 7.0.2

This topic contains issues and major changes that might affect your installation.

- “Changes in this release provided in Upgrade Diff report” on page 311

Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

See also

- “Support” on page 21

Improvements and general issues for ContactManager 7.0.2

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

| ID | Description |
|---------------------------|---|
| Batch Process | |
| PL-18232 | It is now possible to extend any WorkQueue subclass exposed in ContactManager. |
| PL-18404 | Removed the BatchServer configuration parameter from config.xml. Instead, use the JVM argument or registry element isBatchServer to set the batch server.
Also, added the ability to promote a non-batch server to be the batch server on the ClusterInfo page. If a non-batch server is found in the cluster, you can click PromoteToBatch for that server to make that particular server the batch server.
Also, added the promoteToBatch operation to the Cluster Management Bean. |
| PL-18637 | The default ContactManager behavior for query builder result sets is to retrieve all entries from the database into the application server. If the result set is quite large, this can cause problems. Now, you can use the following method to set the query retrieve chunk size.
<pre>public final void setChunkingByID(IQueryResult queryResult, int chunkingSize)</pre> |
| Clustering | |
| PL-11554 | Modified how Guidewire applications discover and maintain the batch server in the clustered environment. Prior to this change, the batch server was discovered with the help of a message exchange through the JGroups cluster communication channel. Users starting up their nodes at approximately the same time could end up with two or more batch servers in the cluster if cluster was not completely formed yet.
This changes uses the underlying database to discover and to keep track of the batch server currently active in the cluster. |
| PL-19141 | Improved the usability of the Cluster Info screen on the Server Tools tab. The screen now shows Server ID and host name information. Guidewire also changed the menu slightly. |
| Cognos Integration | |

| | |
|----------|---|
| PL-19226 | The <code>ab cognos</code> command now builds an application-specific Cognos integration ZIP file, with a name of the form <code>ab-cognos-reporting-integration.zip</code> . Each zip file contains application-specific integration scripts, <code>ab-config</code> and <code>ab-import</code> . These scripts no longer require the namespace/application parameter. |
|----------|---|

Command Line Tools

| | |
|----------|---|
| PL-13663 | Removed hard-coded Java heap settings that could cause out-of-memory issues when running certain tools. |
| PL-18225 | Added WSI document literal web services support to Administration command-line tools. |
| PL-18856 | Fixed an issue that involved regenerating the Data Dictionary. Running <code>gwab regen-dictionary</code> did not actually produce the Data Dictionary files. |
| PL-18892 | <p>Guidewire has added the following optional argument to the data dictionary generation tool:</p> <p><code>maxSPVInclusions</code></p> <p>This value of this parameter defines the depth for second pass verification that limits the number of shared sections that are included the verification of PCF types using <code>verify-all</code>. For example:</p> <pre>gwab regen-dictionary -DmaxSPVInclusions=1000</pre> <p>For this case, the second pass compilation of PCF files stops after 1000 permutation of modal PCF files.</p> <p>Note</p> <ul style="list-style-type: none"> • If you do not include this option, the dictionary generation tool behaves as before. • Use only positive integer values for the <code>maxSPVInclusions</code> property. |

Configuration Upgrade

| | |
|----------|---|
| PL-12804 | Fixed an issue in which the upgrade tool did not handle rules upgrade properly. |
|----------|---|

Consistency Checker

| | |
|----------|---|
| PL-16199 | One-to-one relationships can have at most one entity on each side of the relationship. Under certain conditions, Guidewire is unable to create a unique index to enforce this. In those cases, Guidewire now creates a consistency check to verify the integrity of the data. |
|----------|---|

Contact Domain

| | |
|----------|--|
| PL-18245 | Fixed an issue with <code>IgnoreProperty</code> used in <code>contact-sync-config.xml</code> that caused it to take effect on all subtypes of <code>Contact</code> , rather than only those specified. This functionality now works properly and allows fields to be ignored on contact synchronization only for the specified subtypes. |
|----------|--|

ContactManager

| | |
|---------|---|
| CTC-957 | The minimum search criteria for an <code>ABPerson</code> entity have been changed. They now require that there be a last name or partial last name specified if a first name or partial first name has been specified in the search. Minimum search criteria for contacts are defined in the <code>ContactManager</code> class <code>gw.plugin.contact.ValidateABContactSearchCriteriaPluginImpl</code> . |
| CTC-967 | There was a problem with setting the <code>appscale</code> attribute for <code>MoneyDataType</code> . The setting, specified in <code>datatypes.xml</code> , was being ignored when a column of type <code>money</code> was retrieved. This setting is now working. The <code>appscale</code> attribute controls the number of digits shown to the right of the decimal point for money or currency in single currency mode. It is similar to the <code>scale</code> attribute. It must be smaller than the <code>scale</code> setting and overrides the <code>scale</code> attribute if defined. |

Database

| | |
|----------|--|
| PL-1541 | <p><code>ContactManager</code> now explicitly names primary key constraints by using the following conventions:</p> <ul style="list-style-type: none"> • Oracle – Primary key constraint on the table <code>PX_CONTACT</code> will be named <code>PK_CONTACT</code>. • All other databases – Primary key constraint on the table <code>PX_CONTACT</code> will be named <code>PX_CONTACT_PK</code>. |
| PL-16199 | One-to-one relationships are required to have at most one entity on each side of the relationship. Under certain conditions, Guidewire is unable to create a unique index to enforce this requirement. In those cases, Guidewire now creates a consistency check to verify the integrity of the data. |
| PL-17234 | <code>ContactManager</code> no longer creates shadow tables—tables whose names start with <code>XXt_</code> and <code>XXtt_</code> —for new databases. If you start the application server in dev mode, <code>ContactManager</code> now creates shadow tables only if you set the <code>server.running.tests</code> system property to <code>true</code> , either explicitly or programmatically. The JVM parameter for starting the application server in dev mode is <code>-Dgw.server.mode=dev</code> . |

| | |
|----------|---|
| PL-17286 | Guidewire provides a newer version of the Oracle JDBC driver in the Guidewire application distribution. There is a manual copy required if you are using an external connection pool instead of the Guidewire-bundled connection pool. In this case, you must copy the new driver from the distribution to the location from which the application server data source will pick up the Oracle JDBC driver. |
| PL-17618 | ContactManager now logs the reason for generating a certain step during a database upgrade to the server console log at the DEBUG level. This change also adds this information to the Upgrade Info screen on the Server Tools tab. |
| PL-18637 | The default ContactManager behavior for query builder result sets is to retrieve all entries from the database into the application server. If the result set is quite large, this behavior can cause problems. Now you can use the following method to set the query retrieve chunk size.
<pre>public final void setChunkingByID(IQueryResult queryResult, int chunkingSize)</pre> |
| PL-18749 | With this release, whenever a database upgrade begins, the upgrade process marks the database with the time the upgrade started and the host and batch server from which it began. At the time the database upgrade succeeds, the upgrade process removes the marker. If a second database upgrade begins before the first upgrade finishes, the second upgrade process detects the marker from the first upgrade process and throws an exception.

A second upgrade process can begin if the first upgrade fails and you attempt a restart without first restoring the database. Alternatively, a second upgrade process can begin if a second batch server begins a database upgrade before the first upgrade process completes successfully. |
| PL-19421 | Fixed an issue on SQL Server that caused a stack trace overflow if both of the following were true: <ul style="list-style-type: none"> • There was a connection problem during application start-up. • Configuration parameter MigrateToLargeIDsAndDatetime2 was set to false. |

Document Management

| | |
|----------|--|
| PL-18360 | A new version of the Guidewire Document Assistant ActiveX control will be downloaded to the client browser. |
| PL-18432 | Improved performance for retrieval of extremely large documents through the ActiveX Guidewire Document Assistant. |
| PL-18653 | Guidewire has upgraded the BFO PDF library to version 2.11.20. Refer to the following web site for more details on the changes in this version:
http://bfo.com/viewtext/products/graph/docs/CHANGELOG.txt?title=Big+Faceless+Graph+Library+Changelog |
| PL-18704 | Microsoft Word field forms—text fill-in fields—are now restored correctly while generating a document from a Microsoft Word Template. |

Entities/Metadata

| | |
|----------|--|
| PL-14982 | Modified the Studio Tools → Verify... feature so that it now reports errors and warnings during verification of entity names. |
| PL-18162 | Guidewire has added a validator that prevents you from attempting to denormalize a localized column. Guidewire does not support localized columns for search denorm columns. |
| PL-18266 | Modified the entity type loader to dynamically generate interfaces for the following: <ul style="list-style-type: none"> • Entities that do not have a class • Entities that are extended with a delegate or Java interface that the compile-time backing class does not extend The type loader now uses this runtime interface to create array instances. The runtime impl class now implements this dynamically generated interface as well. |
| PL-18469 | Added a new attribute called <code>typelistTableName</code> to entity types. Use this attribute for non-final entities to specify the name of the corresponding subtype typelist table. If not specified, ContactManager uses the name of the entity as the subtype typelist table name. This capability is useful if an entity name is too long to become a valid typelist table name. |
| PL-19655 | Fixed a problem with columns defined as <code>oneToone</code> that disregarded <code>nullok = false</code> . Now, ContactManager prevents you from committing empty <code>oneToone</code> columns that do not allow nulls. |

- PL-20217 There was a problem with setting the appscale attribute for MoneyDataType. The feature worked for currencyamount columns, but not for money columns. The setting, specified in datatypes.xml, was being ignored when a column of type money was retrieved. This setting is now working.
- The appscale attribute controls the number of digits shown to the right of the decimal point for money or currency in single currency mode. It is similar to the scale attribute. It must be smaller than the scale setting and overrides the scale attribute if defined.

Geocoding/Proximity Search

- PL-1461 Two parameters have been added to the plugin registry for the Geocode plugin. The geocodeDirectionsCulture parameter specifies the locale for geocoded addresses and routing instructions returned from a geocoding and routing service. For example, use the locale code ja-JP for addresses and instructions for Japan. The imageryCulture parameter specifies the language for map imagery. For example, use the language code ja for maps labeled in Japanese.
- PL-18619 Proximity search works around some bad execution plans by disabling index fast full scan and hash join if executing on Oracle. New configuration parameters DisableIndexFastFullScanForProximitySearch and DisableHashJoinForProximitySearch control the workaround. The default value for these parameters is false. These parameters have no effect on databases other than Oracle.

Global Cache

- PL-18322 Fixed an issue with method ProximitySearchQueryUtils.filterWithinRadiusLatLong that did not properly return all the requested points. This issue affected radius searches with a smaller radius value than the number of desired results returned.

Internal Tools/Server Tools Pages

- PL-18017 Guidewire has made the **Internal Tools** tab available in test mode.
- PL-18839 Guidewire has significantly improved the performance of the GW Profiler.
- PL-18845 Modified the **Batch Process Info** screen on the **Server Tools** tab to enable you to download detailed records for any particular batch process for a specific date range. Clicking Download now opens a screen in which you can specify the date range to download for a given batch process.

Localization

- PL-18490 Modified localization.xml to let the thousandsSymbol attribute of the <NumberFormat> element type accept a non-breaking space character as the thousands separator. For example:
- ```
<NumberFormat thousandsSymbol=" ">
```

### Logging

- PL-14722 Added the following new configuration parameters that configure application logging:
- LoggerCategorySource
  - LoggersShowLog4j
- PL-18745 Added support for two new Log4j MDC (Mapped Diagnostic Contexts) keys to include information about the current user in log messages. To use them, include a sequence conforming to the following string in your ConversionPattern in logging.properties:
- ```
%-<LL>.<HH>X{user | userName | userID}
```
- The parameters have the following meanings:
- <LL> – The minimum size of the field. If the actual value is shorter, the user name gets padded with spaces on the right.
 - <HH> – The maximum size of the field. If the actual value is longer, the user name gets truncated from the left.
 - user – Prints the user's internal numeric ID number, such as 4231341234. This parameter was available in previous releases and remains unchanged for compatibility.
 - userName – Prints the user's real-world name, such as John Smith.
 - userID – Prints the user's username in the system, such as jsmith.
- For example:
- ```
%-16.16X{userName}
```



PL-20907	<p>Guidewire updated the logging API between the following releases:</p> <ul style="list-style-type: none"> <li>• BillingCenter 7.0.1 and 7.0.2</li> <li>• ClaimCenter 7.0.1 and 7.0.2</li> <li>• PolicyCenter 7.0.3 and 7.0.4</li> </ul> <p>Guidewire has created the following Knowledge Base article that describes the changes made to the logging API: <i>Changes to the logging API during upgrade to BillingCenter 7.0.2, ClaimCenter 7.0.2, and PolicyCenter 7.0.4 or later versions</i></p> <p>Review this article if you are performing an upgrade, in order to update your configuration to the new logging API. If necessary, contact Guidewire Support for a copy.</p>
----------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### Messaging

PL-18678	<p>The <code>MessageSenderRunnable.run</code> method now includes the <code>messageID</code> in the log. The log entry will look similar to the following:</p> <pre>MW.MessageSenderRunnable.run (dest destination_id):     Entering run() for messageId message_id</pre>
PL-19136	<p>Fixed an issue in which a race condition in a non-clustered server could lead to inconsistent results or runtime exceptions or both when modifying entities within a message transport.</p>
PL-19149	<p>After locking out a user who has reached the maximum number of allowed login failures, <code>ContactManager</code> no longer ignores generated events. Instead, <code>ContactManager</code> executes the transaction with a service token having a super user. As a result, <code>ContactManager</code> now generates <code>UserChanged</code> events properly.</p>

#### Miscellaneous

PL-3760	<p>Guidewire has made the following modification to <code>config.xml</code>:</p> <ul style="list-style-type: none"> <li>• Removed element <code>&lt;security sessiontimeoutsecs="10800"/&gt;</code></li> <li>• Added configuration parameter <code>&lt;param name="SessionTimeoutSecs" value="10800"/&gt;</code> as its replacement</li> </ul> <p><code>ContactManager</code> performs an automatic upgrade of <code>config.xml</code> for this change.</p>
PL-15462	<p>Guidewire applications provide several configuration parameters that accept a comma-separated list of values. Now, <code>ContactManager</code> parses a comma-separated list of values according to CSV format and trims each value automatically. This change enables you to use spaces, tabs, and new lines for more readability while specifying such a value.</p> <p>For example:</p> <ul style="list-style-type: none"> <li>• A value that before the fix had to be without spaces – <code>AUTO,PR,GL,TRAV</code></li> <li>• The same value after the fix can use spaces – <code>AUTO, PR, GL, TRAV</code></li> </ul> <p>The following is a more technical description of the format:</p> <ul style="list-style-type: none"> <li>• Fields must be separated with commas.</li> <li>• Leading and trailing spaces are ignored unless the field is delimited with double-quotes. In that case, the white spaces are preserved.</li> <li>• Embedded comma – You must delimit the field with double-quotes, with the comma inside the double quotes.</li> <li>• Embedded double-quotes – You must double embedded double-quote characters, and you must delimit the field with double-quotes.</li> <li>• Embedded line-breaks – You must surround the field with double-quotes.</li> <li>• Always Delimiting – You can always delimit a field with double-quotes. If not strictly needed, the delimiters will be parsed and discarded by the reading applications.</li> </ul>
PL-15914	<p>Guidewire has added <b>Save</b> and <b>Cancel</b> buttons on the <b>Server Tools Management Beans→Guidewire Managed Bean Properties</b> screen. You can use these buttons to save or cancel a change to a bean property value. The buttons become active after you edit an editable property.</p>
PL-18245	<p>Fixed an issue with <code>IgnoreProperty</code> used in <code>contact-sync-config.xml</code> that caused it to take effect on all subtypes of <code>Contact</code>, rather than only those specified. This functionality now works properly and allows fields to be ignored on contact synchronization only for the specified subtypes.</p>
PL-19311	<p>Guidewire has upgraded the Joda-Time third-party library version to 2.0.</p>

#### Page Configuration

PL-15548	<p>Printing a second-level list view no longer results in missing items, nor does it cause items to print more than once.</p>
----------	-------------------------------------------------------------------------------------------------------------------------------

PL-18303	Adding multiple entries in list views for custom entity types no longer causes errors.
PL-19602	Added the ability to retain the scroll position after clicking a pull-right menu item. This fix adds the <code>retainScrollPosition</code> property to the <code>MenuItem</code> widgets.
<b>Persistence</b>	
PL-18162	Guidewire has added a validator that prevents you from attempting to denormalize a localized column. Guidewire does not support localized columns for search denorm columns.
<b>PL Services</b>	
PL-11554	Modified how Guidewire applications discover and maintain the batch server in the clustered environment. Prior to this change, the batch server was discovered with the help of a message exchange through the JGroups cluster communication channel. Users starting up their nodes at approximately the same time could end up with two or more batch servers in the cluster if the cluster was not completely formed yet. This change uses the underlying database to discover and keep track of the batch server that is currently active in the cluster.
PL-18232	It is now possible to extend any <code>WorkQueue</code> subclass exposed in <code>ContactManager</code> .
<b>Queries</b>	
PL-17541	Modified <code>GWDBFunctionEnhancement.gsx</code> and added support for <code>DBFunction</code> calls of <code>MIN</code> and <code>MAX</code> on <code>java.util.Date</code> .
PL-19590	Fixed an issue that caused an error in the <code>ContactManager</code> user interface by a query that contained a column definition twice.
<b>Rules Editor</b>	
PL-19466	Fixed a problem that occurred when you dragged a rule to the top of a rule set. Guidewire Studio incorrectly overwrote the first rule instead of inserting the lower order rule above it. Now Studio inserts the lower order rule correctly above the top rule.
PL-19510	Fixed an issue that caused errors in the Guidewire Studio Find in Path functionality when searching rule resources if the resources contained files that were not proper Gosu rule resources. Proper Gosu rule resources contain Gosu code and end with the file extensions <code>.grs</code> or <code>.gr</code> . Now, Find in Path displays non-Gosu rule resources as errors without affecting search results for valid resources.
<b>Studio</b>	
PL-19568	Guidewire Studio now supports 64-bit Java Virtual Machines (JVMs).
<b>Studio IDE</b>	
PL-13352	Fixed an issue in which Studio ignored the formatting settings for <code>String</code> literals set in Studio <b>Tools</b> → <b>Options</b> → <b>Colors and Fonts</b> .
PL-15219	Fixed an issue with the <code>TemplateInputWidget</code> PCF file in which autocomplete inserted a value that appeared to be correct, but failed validation.
PL-15887	Fixed an incorrect message that Studio showed in the <b>Verify Result</b> pane if verifying a valid entity name in the <b>Entity Names</b> editor.
PL-18203	Fixed an issue in which the Studio debugger step-over functionality stopped on Guidewire internal code in between executing Rule Condition and Rule Action code.
PL-18671	Guidewire has modified Studio behavior in regards to readonly mode as follows: <ul style="list-style-type: none"> <li>• Studio now shows a padlock button on the status bar that is visible only if it is in readonly mode. If you click the button, Studio displays a modal message box indicating the reasons why it is in readonly mode.</li> <li>• Studio disables the <b>Save</b> button any time that it is in readonly mode.</li> <li>• Studio changes the <b>Save</b> button tooltip in readonly mode to show the reason that save is not active in this mode. This message is the same one that Studio shows if you click the padlock icon on the status bar.</li> </ul>
PL-18905	Fixed an issue in which it was possible to edit configuration files even if Studio was in readonly mode.
<b>Utilities</b>	

PL-11681	<p>The Gosu StringUtil class used many Perl 5, version 3, regular-expression, syntax-compatible functions to search for and replace strings. These functions were implemented by the Apache/Jakarta ORO library, which has been retired. See <a href="http://jakarta.apache.org/oro/">http://jakarta.apache.org/oro/</a> for details.</p> <p>The same functionality for regular-expression, syntax-compatible search and replace of strings is now available in the java.util.regex package and the java.lang.String class.</p> <p>Before this change, you used the following coding pattern:</p> <pre>StringUtil.substitute(inputString, "s/"+ regexString + "/" +     replacementString + "/g")</pre> <p>Now, instead, you use the following coding pattern:</p> <pre>var pattern = java.util.regex.Pattern.compile(regexString) var patchedString = pattern.matcher(inputString).replaceAll(replacementString)</pre> <p>Simpler search and replace functions are also available, such as:</p> <pre>java.lang.String.replace(String, String).</pre>
PL-15733	Guidewire has updated the Google Guava library to release 10.
<b>Web</b>	
PL-15548	Printing a second-level list view no longer results in missing items, nor does it cause items to print more than once.
PL-18303	Removed errors caused while adding multiple entries in list views for custom entity types.
PL-19454	Fixed an issue that occurred while using Internet Explorer 9 in which clicking a drop-down list did not render the list properly.
PL-19602	Added the ability to retain the scroll position after clicking a pull-right menu item. This fix adds the retainScrollPosition property to the MenuItem widgets.
<b>Web Services</b>	
PL-18361	ContactManager now provides the ability to implement a pre-existing WSDL in Gosu for a WSI web service.
PL-18450	ContactManager now consumes MTOM-enabled WS-I web services. MTOM is the W3C Message Transmission Optimization Mechanism that efficiently sends binary data to and from web services. You do not need to do anything to take advantage of this new feature.
PL-18589	Added the logging category XML .Request for WSI web services to log each request. Each request will be logged at the DEBUG level. This logging includes the connecting address and user, if available, as well as the request qname, which is unique for each operation.
<b>Web – UI/Runtime</b>	
PL-19646	Fixed an issue preventing the Guidewire application shortcut keys from working as intended. Previously, the keyboard responded only to the shortcut keys of the browser. Now the keyboard responds to the Guidewire application shortcut keys.
<b>Work Queues</b>	
PL-18675	Fixed several issues that involved the <b>Work Queue Info</b> screen on the <b>Server Tools</b> tab.
PL-19003	You can now start or stop work queues from IMaintenanceToolsAPI.
PL-19004	Added the attribute orphansfirst to a worker, thereby making it possible to specify that ContactManager is to process orphans before new items on this worker.
PL-19059	The <b>Work Queue Info</b> screen on the <b>Server Tools</b> tab now reports the last time that ContactManager processed a work item, showing last Success time or last Exception time. If the time was for an exception, the <b>Work Queue Info</b> screen also reports the number of consecutive work items that resulted in an exception. The screen shows two other dates as well. The Last Notification time is the last time the worker woke up. If it found work, ContactManager also updates the Last Batch Found time.

## Known issues and limitations for ContactManager 7.0.2

This section describes known issues with this release of Guidewire ContactManager.

- “ContactManager known issues for release 7.0.2” on page 444
- “Studio/Platform known issues for ContactManager 7.0.2” on page 444

**Note:** For maintenance releases, Guidewire often defers fixing configuration issues if they require merging files during the upgrade. Workarounds to many of these issues are listed in the following sections. The goal of this policy is to make upgrades as straightforward as possible.

### ContactManager known issues for release 7.0.2

#### RelatedABContactSearchCriteria does not work for related contact searches (CTC-960)

**Issue** – Using the `RelatedABContactSearchCriteria` entity and the `CriteriaDef` elements that are defined for it in `search-config.xml` is not supported in ContactManager 7.0.2.

**Workaround** – Do not use them. Guidewire is aware of this issue.

#### Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

**Issue** – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not check to make sure the relationship does not already exist. Therefore, ContactManager can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

**Workaround** – In the base configurations of d ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one `ContactContact` entity in this case.

### Studio/Platform known issues for ContactManager 7.0.2

#### Issues with Internet Explorer 9

**Issue** – If you are using the Internet Explorer 9 browser, it is possible to see issues such as screen flickering or an incorrect tab order for fields. According to public reports, Internet Explorer 9 exhibits these and other issues with a variety of web sites and web applications.

**Workaround** – Because this is the behavior of the Internet Explorer 9 rendering engine, Guidewire cannot address these issues. However, there are reports of an Internet Explorer 9 workaround that reduces these issues. Change the new Internet Explorer 9 **Accelerated Graphics** settings on the **Advanced** tab of the **Internet Options** dialog.

#### First time you click the arrow of the typekey input, the drop-down menu does not open (PL-10134)

**Issue** – The drop-down menu does not open on the first click of the arrow on a typekey input. Instead, the help text opens.

**Workaround** – Turn off help text on focus by setting `InputHlepTextOnFocus` to `false` in the `config.xml` file. After you do that, the help text shows only if you mouse over the input, and it does not interfere with opening a drop-down menu.

#### XML API upgrade feature missing from documentation (PL-10257)

**Issue** – The *Integration Guide* describes a new set of XML APIs based on the `XmlElement` class. (Legacy APIs are based on the `XMLNode` class.) You can continue to use the legacy APIs. However, the *Integration Guide* omits mentioning an additional upgrade-specific feature.

**Workaround** – For backwards compatibility only, you can import an XML schema into the Gosu type system using the legacy XML system by following these instructions:

1. Copy:

```
ContactManager/modules/pl/config/registry/compatibility-xsd.xml
```

To:

```
ContactManager/modules/configuration/config/registry/compatibility-xsd.xml
```

2. Add an entry for your schema. Set the value of the namespace attribute to the Gosu package name of the schema. For example, if the schema is in the package location `my.package` and is called `myschema.xsd`, set the value of namespace to `my.package.myschema`.

### Studio Rules do not use correct capitalization for root object's name (PL-10740)

**Issue** – Rule set root objects are not named with first letter lowercased.

**Workaround** – The rules engine issues a warning if the correct case for objects is not being used.

### Countries configured in zone-config.xml still generate a warning during regen-dictionary even when zone data is loaded for all these countries (PL-11947)

**Issue** – Countries configured in `zone-config.xml` still generate a warning during regen-dictionary even when zone data is loaded for all of these countries.

**Workaround** – Warning message is created in error and can safely be ignored.

### User interface cannot handle starting multiple instances of a batch process (PL-12372)

**Issue** – The user interface cannot handle starting multiple instances of a batch process.

**Workaround** – To execute multiple instances of a batch process, start them from the command line. Also, to allow multiple instances to be run simultaneously, you must ensure that the `BatchProcess.isExclusive()` method returns `false`.

### Type system refresh after PCF page title change does not update corresponding menu label (PL-13057)

**Issue** – The type system refresh after a PCF page title change does not update the corresponding menu label.

**Workaround** – After updating a page title, restart the server to refresh menu labels and avoid null pointer exceptions due to stale references.

### Length limitation on entity localization table names (PL-13360)

**Issue** – Entity localization table names have a shorter, 16-character limit than other entity tables. If the localization table name exceeds the 16-character limit, the error message indicates incorrectly that 18 characters are allowed.

**Workaround** – Ensure that the localization `tableName` property specified in the entity extension file is less than 16 characters.

### US-Locations.txt file with the US geodata from GreatData has special characters that cause validation problems with United States Postal Service (USPS) data (PL-13384)

**Issue** – The `US-Locations.txt` file contains information that does not conform to United States Postal Service (USPS) standards for bulk mailings.

**Workaround** – The provided `US-Locations.txt` file is intended only for use in geocoding to identify addresses for a location. You can edit the `US-Locations.txt` file to conform to your particular address standards, and then import that version of the file instead.

### GX models that reference virtual fields and enhancements throw null pointers if null (PL-13560)

**Issue** – The GX models that reference virtual fields and enhancements throw null pointers when these fields and enhancements are null.

**Workaround** – Include null checks and error handling to prevent referenced virtual fields or enhancements that are null from causing null pointer exceptions.

### Sending email with file attachment with unicode filename is not correctly handed over to the mail server (PL-13582)

**Issue** – An email with a file attachment that has a unicode file name is not sent to the mail server correctly.

**Workaround** – Use Latin characters for file names on attached files.

### GX model generated XSD cannot be parsed by JAXB (PL-13598)

**Issue** – XSD generated by the GX model cannot be parsed by JAXB.

**Workaround** – Add JAXB annotation elements to the XSD to specify the necessary metadata, such as class names, to enable JAXB to generate the Java class files. Contact Guidewire Support for a sample XSD file that is annotated this way.

### Cannot make a field from a delegate into a localized column (PL-13761)

**Issue** – You cannot make a field from a delegate into a localized column.

**Workaround** – Move the column to be localized off the delegate and onto each of the implementing entities. Then, to make the column appear as though it exists on the delegate, define an enhancement property on the delegate that *delegates* to the appropriate column.

### Problem with regen-java-api command and JAR files (PL-16351)

**Issue** – If you run the `ContactManager/bin/gwab regen-java-api` command, ContactManager creates a `ContactManager/java-api/lib` directory and puts JAR files for Java development in this directory. However, ContactManager does not always perform this task correctly, resulting in some of the generated JAR files' not being copied to this directory.

**Workaround** – For missing library files, copy or import into your Java development environment the missing library files from the following directory:

```
ContactManager/admin/lib
```

Copy them into the following directory:

```
ContactManager/java-api/lib
```

### Renaming method or property throws ParseResultsException (PL-16633)

**Issue** – If you rename a property or method or you change a method signature, and a workflow references that property or method in a Gosu field, ContactManager throws `ParseResultsException`. This is the intended behavior.

**Workaround** – Restart the workflow engine. To do so:

1. Log into ContactManager using an administrative account.
2. Access **Internal Tools**→**Reload**.

Click **Reload Workflow Engine**.

## Guidewire ContactManager 7.0.1 release notes

These release notes contain the following:

- “Release Information for ContactManager 7.0.1” on page 447
- “Installing ContactManager 7.0.1” on page 447
- “Support” on page 21
- “Issues and major changes for ContactManager 7.0.1” on page 447
- “Improvements and general issues for ContactManager 7.0.1” on page 449
- “Known issues and limitations for ContactManager 7.0.1” on page 457

### Release Information for ContactManager 7.0.1

These release notes apply only to this release of Guidewire ContactManager.

---

**IMPORTANT** If you are upgrading and you skipped one or more releases of ContactManager, be sure to read the release notes for those releases to learn about changes and fixed issues. See “Installing ContactManager 7.0.1” on page 447.

---

This release of Guidewire ContactManager is 7.0.1.

- If you are integrating with Guidewire BillingCenter, this version of ContactManager requires BillingCenter 7.0.0 or later.
- If you are integrating with Guidewire PolicyCenter, this version of ContactManager requires PolicyCenter 7.0.1 or later.
- If you are integrating with Guidewire ClaimCenter, this version of ContactManager requires ClaimCenter 7.0.0 or later.

### Installing ContactManager 7.0.1

For general installation information, see “Installing ContactManager” on page 55.

For any versions prior to 7.0.1 that you have skipped, see prior ClaimCenter release notes, which include a section for ContactManager release notes. For example, see the release note archive in the *New and Changed Guide*.

### Support

For assistance, visit the Guidewire Community.

#### Guidewire customers

<https://community.guidewire.com>

#### Guidewire partners

<https://partner.guidewire.com>

### Issues and major changes for ContactManager 7.0.1

This topic contains issues and major changes that might affect your installation.

- “ContactManager Integration with Core Applications” on page 448
- “Geocoding Using Bing and MapPoint (PL-16708)” on page 448
- “New web services APIs return current AddressBookUID for merged contact (CTC-588)” on page 448
- “Provide web service for validating contacts for creation (CTC-603)” on page 448
- “Make ValidateABContactCreationPlugin return the error message (CTC-611)” on page 448
- “Make ValidateABContactSearchCriteriaPlugin return the error message (CTC-612)” on page 449
- “Changes in this release provided in Upgrade Diff report” on page 311



## ContactManager Integration with Core Applications

This release of ContactManager provides integrations with Guidewire ClaimCenter, PolicyCenter, and BillingCenter. For more information, see “Integrating ContactManager with Guidewire core applications” on page 59.

## Geocoding Using Bing and MapPoint (PL-16708)

Guidewire has added an implementation of the `GeocodePlugin` that connects to the Microsoft Bing Maps Geocode Service. The Bing Maps plugin implementation replaces the Microsoft MapPoint implementation.

Guidewire has deprecated the MapPoint implementation. Microsoft announced plans to retire the MapPoint web service effective November 18, 2011. If you currently use geocoding features and the MapPoint plugin, you must migrate from MapPoint to Bing Maps. Otherwise, geocoding features in the application will not function.

## New web services APIs return current AddressBookUID for merged contact (CTC-588)

When a contact is retired as part of a merge operation, ContactManager now keeps track of the `LinkID` of the contact that replaced it. There is no safe ordering on merge messages. Therefore, a subsequent merge might occur before the core application queries ContactManager to retrieve the `LinkID` for the replacement contact. ContactManager now provides APIs that enable a core application to discover the current `LinkID` in ContactManager that replaces a `LinkID` that the application has to a merged contact.

The APIs that ContactManager supplies for this purpose are:

- `ABContactAPI.getReplacementContact(contactLinkID : String) : String`
- `ABContact.getReplacementContact() : ABContact`

To access this method, you must fetch updates for the ContactManager web service `ABContactAPI` in Guidewire Studio for your core application, as follows:

1. Ensure that ContactManager is running.
2. Start Guidewire Studio for your core application.
3. In the **Project** window, navigate to **configuration**→**Classes**→**ws**→**remote**→**gw**→**webservice**→**ab**→**ab700**.
4. In the editor on the right, click the following resource in the **Resources** pane:

```
${ab}/ws/gw/webservice/ab/ab700/abcontactapi/ABContactAPI?wsdl
```

5. Click **Fetch Updates** to get the latest updates to `ABContactAPI`. If you see a message asking if you want to create a copy of the resource, click **Yes**.

## Provide web service for validating contacts for creation (CTC-603)

The ContactManager web service `ABContactAPI` has a new method that determines if the specified contact can be created.

The method, `ABContactAPI.validateCreateContact`, calls `ValidateABContactCreationPlugin` to see if the contact has enough data specified to allow it to be created.

The method returns an `ABContactAPIValidateCreateContactResult` object, which is a Gosu version of the Java object `ValidateABContactCreationPluginResult` that the `ValidateABContactCreationPlugin` plugin returns. For more information, see “`ValidateABContactCreationPlugin` plugin interface” on page 297.

To use this method, you must fetch updates for the ContactManager web service `ABContactAPI` in Guidewire Studio for your core application, as described previously for CTC-588. See “New web services APIs return current AddressBookUID for merged contact (CTC-588)” on page 448.

## Make ValidateABContactCreationPlugin return the error message (CTC-611)

`ValidateABContactCreationPluginImpl.validateCanCreate` used to return a `Boolean` indicating whether validation passed. Now it returns a `ValidateABContactCreationPluginResult` object that has the following methods:

- `boolean isValid();`
- `String getErrorMessage();`



The code in the class did not change much. If you changed the `validateCanCreate` method itself, you must apply these changes to the new private method `canCreate`. If you changed code elsewhere in the class, you must re-apply those changes.

For more information on this class, see “`ValidateABContactCreationPlugin` plugin interface” on page 297.

## Make `ValidateABContactSearchCriteriaPlugin` return the error message (CTC-612)

`ValidateABContactSearchCriteriaPluginImpl.validateCanCreate` used to return a `Boolean` indicating whether validation passed. Now it returns a `ValidateABContactSearchCriteriaPluginResult` object that has the following methods:

- `boolean isValid();`
- `String getErrorMessage();`

The code in the class did not change much. If you changed the `validateCanCreate` method itself, you must apply these changes to the new private method `canCreate`. If you changed code elsewhere in the class, you must re-apply those changes.

## Changes in this release provided in Upgrade Diff report

Guidewire provides a report detailing certain differences between the current release and your prior release. This report describes changes in display keys, entities, typelists, and the Gosu API. To obtain your custom Upgrade Diff report, visit the Guidewire Community.

### See also

- “Support” on page 21

## Improvements and general issues for ContactManager 7.0.1

This topic describes improvements and issues corrected in this release. Guidewire attempts to provide information for issues of primary importance to our customers. This list is not intended to be comprehensive.

ID	Description
ContactManager	
CTC-588	<p>When a contact is retired as part of a merge operation, ContactManager now keeps track of the LinkID of the contact that replaced it. There is no safe ordering on merge messages. Therefore, it is possible that a subsequent merge has occurred by the time the core application queries ContactManager to retrieve the LinkID for the replacement contact. ContactManager now provides APIs to the core applications so they can discover the current LinkID for a LinkID they have to a contact that has been merged.</p> <p>The APIs that ContactManager supplies for this purpose are:</p> <ul style="list-style-type: none"> <li>• <code>ABContactAPI.getReplacementContact(contactLinkID : String) : String</code></li> <li>• <code>ABContact.getReplacementContact() : ABContact</code></li> </ul> <p>To access this method, you must fetch updates for the ContactManager web service <code>ABContactAPI</code> in Guidewire Studio for your core application, as follows:</p> <ol style="list-style-type: none"> <li>1. Ensure that ContactManager is running.</li> <li>2. Start Guidewire Studio for your core application.</li> <li>3. In the <b>Resources</b> pane on the left, navigate to <code>configuration→Classes→wsi→remote→gw→webservice→ab→ab700</code>.</li> <li>4. In the editor on the right, click the following resource in the <b>Resources</b> pane: <code>\${ab}/ws/gw/webservice/ab/ab700/abcontactapi/ABContactAPI?wsdl</code></li> <li>5. Click <b>Fetch Updates</b> to get the latest updates to <code>ABContactAPI</code>. If you see a message asking if you want to create a copy of the resource, click <b>Yes</b>.</li> </ol>
CTC-596	<p>In some cases, the Duplicate Contact Finder batch process was picking the wrong contact as the contact to be kept. It now consistently picks the older of two contacts as the kept contact.</p>

CTC-599	Importing sample data from a locale other than en_US, en_GB, en_AU, and en_NZ, such as ja_JP, was throwing an exception when the withTaxId method was called. This method has been changed to return an empty string for locales other than these four locales.
CTC-603	<p>The ContactManager web service ABContactAPI has a new method that determines if the specified contact can be created.</p> <p>The method, ABContactAPI.validateCreateContact, calls ValidateABContactCreationPlugin to see if the contact has enough data specified to allow it to be created.</p> <p>The method returns an ABContactAPIValidateCreateContactResult object, which is a Gosu version of the Java object ValidateABContactCreationPluginResult that the ValidateABContactCreationPlugin plugin returns. For more information, see “ValidateABContactCreationPlugin plugin interface” on page 297.</p> <p>To use this method, you must fetch updates for the ContactManager web service ABContactAPI in Guidewire Studio for your core application, as described previously for CTC-588.</p>
CTC-606	The ContactManager AddressBuilders.withBatchGeocode(true) method was not setting the Address.Geocode field to true for sample code in the database. This method is working correctly now.
CTC-609	Incorrect permission check in ABContactAPI.retrieveRelatedContacts. This method used perm.Contact.viewab. Instead, the correct permission check is perm.ABContact.view because the method deals with ABContact objects and not Contact objects. It now uses the correct permission check expression.
CTC-611	<p>ValidateABContactCreationPluginImpl.validateCanCreate used to return a Boolean indicating whether validation passed. Now it returns a ValidateABContactCreationPluginResult object that has the following methods:</p> <ul style="list-style-type: none"> <li>• boolean isValid();</li> <li>• String getErrorMessage();</li> </ul> <p>The code in the class did not change much. If you changed the validateCanCreate method itself, you must apply these changes to the new private method canCreate. If you changed code elsewhere in the class, you must re-apply those changes.</p> <p>For more information on this class, see “ValidateABContactCreationPlugin plugin interface” on page 297.</p>
CTC-612	<p>ValidateABContactSearchCriteriaPluginImpl.validateCanCreate used to return a Boolean indicating whether validation passed. Now it returns a ValidateABContactSearchCriteriaPluginResult object that has the following methods:</p> <ul style="list-style-type: none"> <li>• boolean isValid();</li> <li>• String getErrorMessage();</li> </ul> <p>The code in the class did not change much. If you changed the validateCanCreate method itself, you must apply these changes to the new private method canCreate. If you changed code elsewhere in the class, you must re-apply those changes.</p>
Assignment	
PL-13175	Delegated the UserRoleAssignment entity out to a UserRoleAssignmentDelegate and removed the UserRoleAssignment entity at the platform level. All Guidewire applications now own their own UserRoleAssignment entity and use entity delegation to UserRoleAssignmentDelegate.
PL-18196	Fixed assignGroupByRoundRobin to manage GroupAssignmentState with proper set of keys including intended GroupType to assign a group properly.
PL-18397	Fixed an issue that occurred during the creation of an exposure at the end of FNOL wizard if you were using assignGroupByRoundRobin assignment.
Batch Processes	
PL-13912	Guidewire has added the ability to set an env attribute on the <ProcessSchedule> element in scheduler-config.xml to specify the schedule to run in a certain environment. As a consequence, you can now have different results for batch processing based on environment. By default, Guidewire does not set the env attribute.
PL-15657	Added <b>Last Run Status</b> column to the <b>Processes</b> table of the <b>Batch Process Info</b> screen. This column allows a user to see if the last run of this batch task completed successfully or failed.

The possible values are:

- **Completed** – The last run has completed successfully.
- **Not available** – This task has not ran yet or running right now.
- **Failed/Interrupted** –The last run has failed or was interrupted. See the history of the process for more details.

Clustering	
PL-10224	In addition to improvements to JGroups implementation of UNICAST protocol, this fix also introduced a new merge protocol, MERGE4. This is not set in the default protocol stack, in which MERGEFAST is set.
PL-17295	The valid range for multicast address is between 224.0.0.0 and 239.255.255.255. This change verifies the validity of the multicast address before attempting to join the cluster. If the address is invalid, ContactManager throws the following exception: <code>"Invalid multicast address " + multicastAddress + ".  Valid range is between 224.0.0.0 and 239.255.255.255."</code>
PL-17297	Guidewire has upgraded JGroups to the latest version, which is 2.12.1.
PL-17903	Guidewire now requires a unique server ID for each node in a cluster. The application verifies its server ID for uniqueness across all other cluster members as the node starts.
Code Utilities	
PL-18054	If you have used the <code>{x}</code> substitution in document templates, then you need to revert those uses back to the <code>&lt;%= ... %&gt;</code> syntax.
Configuration Upgrade	
PL-18021	Fixed an issue with how the Upgrade tool automatically handled rule upgrades.
Database	
PL-17368	Database upgrade no longer runs certain statistics commands automatically. Database upgrade no longer includes the following: <ul style="list-style-type: none"> <li>• Upgrade does not update statistics on indexes as both SQL Server and Oracle have options to automatically create statistics on indexes during database creation.</li> <li>• Upgrade does not update statistics on tables and columns for new database.</li> </ul>
PL-17388	Modified the length of the columns created to support linguistic searching, the <i>DENORM</i> columns, reducing them in size from twice the original column's size to exactly the original column's size
PL-18012	Previously, Guidewire prohibited you from connecting a development server to a production database, or connecting a production server to a development database. Guidewire now permits this kind of connection (merely generating a warning) if using the H2 database.
PL-18142	Fixed an issue in the Leap Year Days calculation that caused an infinite loop if times in the start and end of the DateRange did not match.
Database Upgrade	
PL-17732	Fixed performance issues with the upgrade process.
PL-17849	Guidewire has added a <code>DateFromDatetime</code> method to the <code>BeforeUpgradeDBFunction</code> utility class, which is available only in the context of upgrade queries. It is available only in restrictions, not in the query column list. Certain Guidewire applications need this for use in an upgrade trigger. In a <code>BeforeUpgrade</code> query, you can use this function in a <code>SELECT</code> statement, and it can take a nested function as an argument
Datamodel	
PL-17540	It is now possible to add an event to an entity through an extension, even if the base configuration definition of the entity contains one or more events.
PL-7602	Enhanced GosuDoc to catch all throwables thrown when attempting to load a type. Previously, it only it only caught Runtime Exceptions.
PL-15360	Fixed an issue in which Guidewire allowed <code>@WebServices</code> annotations at the function level in some cases. Now, you can use this annotation at the class level only.

PL-15806	Fixed an issue in which using the == operator in an overridden equals method caused a stack overflow.
Entities/Metadata	
PL-10608	Guidewire now flags as an error in Studio—and as a warning at server start-up—if you set a non-queryable path on the <b>Entity Path</b> field in the Entity Names editor.
PL-17718	Fixed an issue in which a forward slash (/) in a typecode definition prevented the application server from starting, but Studio did not display an error flag for this condition.
PL-18071	Added support for denormalizing columns across tables to enhance the performance of search queries.
Geocoding/Proximity Search	
PL-16708	Guidewire has added an implementation of the GeocodePlugin that connects to the Microsoft Bing Maps Geocode Service. The Bing Maps plugin implementation replaces the Microsoft MapPoint implementation.
PL-17675	Fixed an issue in which the Bing Geocode plugin implementation was not visible in Studio.
PL-17886	The MapPoint implementation is deprecated by Guidewire with this release. Microsoft announced plans to retire the MapPoint web service, effective November 18, 2011. If you currently use geocoding features and the MapPoint plugin, you must migrate from MapPoint to Bing Maps before November 18, 2011. Otherwise, geocoding features in the application cease to function on November 18, 2011, and afterwards.
Gosu	
PL-16935	Fixed an issue in which ContactManager threw a Null Pointer Exception if you attempted to change Gosu code in Studio and it was connected to the application server. (The issue was specific to functions in code blocks that were called from other PCF functions.)
PL-17478	Modified the Gosu Tester parser to handle scriptability modifiers in the same way that Gosu classes and enhancements do. The Gosu Tester no longer shows the following error message for scriptability modifiers: 'The property [or method], "Xxxx", is not visible under the parser's visibility constraints.'
PL-17489	Fixed an issue in which certain corner cases of conditional variable declarations would fail with a Java byte-code VerifyError if the server was running in debug mode. This issue did not affect servers running in production mode.
PL-18033	Fixed an issue that caused incorrect behavior if you called reverse() on an implicitly-declared List in Gosu. (The issue did not occur with an explicitly declared List.) The issue was that the reverse sort on an implicitly-declared List occurred in-place, meaning that the sort occurred on the original list elements. The sort now occurs on a copy of the list, which is the desired behavior.
PL-18299	Added a preload mechanism to support pre-compilation of Gosu classes, as well as other primary classes in the system. The intent is to make the system more responsive the first time requests are made. To support this, Guidewire has added a Studio <b>Other Resources</b> preload.txt file in which you can add a list of actions to take. The file contains static no-argument method calls, as well as the names of Gosu types to compile to byte-code or the Java types to load. Guidewire also added a new logging category of Server.Preload that provides DEBUG level logging of all actions during server pre-loading of Gosu classes.
GX Tools	
PL-12128	Added the ability for a web service to use types from a different WSDL. For example, <pre>&lt;xs:import namespace="http://example.com/gw/api/test/TheirAPI"   schemaLocation="../../../wsi/local/gw/api/test/TheirAPI.wsdl" /&gt;</pre>
PL-18297	Restored missing eachException() methods on the Guidewire XML Modeler. Guidewire inadvertently removed these methods in a previous 7.x release.
History	

PL-17720	Guidewire no longer marks the <code>HistoryType</code> typelist as <code>final</code> . Therefore, it is now possible to add new values to this typelist. However, Guidewire made this change to allow the removal of the retrieved <code>HistoryType</code> in ClaimCenter. Guidewire recommends that you continue to add custom history types to the <code>CustomHistoryType</code> typelist, rather than the <code>HistoryType</code> typelist.
Integration	
PL-17479	The base configuration no longer contains a <code>suite-config.xml</code> file that contains default URL values of <code>productname</code> . Instead, this file has entries that are commented out. Guidewire considers any product not configured in <code>suite-config.xml</code> , which is the default for all products, to be non-existent in the suite.
PL-16370	The <code>regen-soap-api</code> command now generates RPC-Encoded and WSI artifacts into different directories.
PL-17945	Guidewire has changed the type of the URL field in the <code>suite-config</code> XSD definition file from <code>xs:anyURI</code> to <code>xs:string</code> .
PL-17946	Guidewire now disallows duplicate entries in <code>suite-config.xml</code> . Guidewire defines duplicate entries as those with the same name, URL, and env values.
Logging	
PL-18234	Improved logging of <code>ConcurrentDataChangeException</code> . Now, the log message shows both of the following: <ul style="list-style-type: none"> <li>The username of the current user, who received the <code>ConcurrentDataChangeException</code></li> <li>The previous user, who made the initial data modification</li> </ul> The log writes the current user's name before the previous user's name in the message.
Messaging	
PL-18211	Fixed an issue in which messages could potentially be sent in incorrect send order if a bundle commit contained multiple event root entities.
Miscellaneous	
PL-13354	Guidewire now certifies Oracle WebLogic for use as an application server for ContactManager. Visit the Guidewire Community and search for knowledge article 1005, "Supported Software Components" for more detail on the exact version that has been certified.
PL-17350	Fixed an issue with <code>regen-soap-api</code> that caused it to not properly generate the <code>template/toolkit-javadoc/gw-ab-plugin</code> directory.
PL-16779	Fixed an issue with <code>regen-soap-api</code> that caused it to fail if specific MQ libraries existed in <code>configuration/plugins/shared/lib</code> or <code>gosu/lib</code> . Executing <code>regen-soap-api</code> with these libraries now logs a warning rather than forcing an outright failure.
PL-17987	Guidewire has removed the extraneous <code>Bundle</code> property on the <code>Bundle</code> object ( <code>Bundle.Bundle</code> ).
Other - PL Services	
PL-14011	The (Server Tools) <b>Management Beans</b> → <b>CurrentUserSessions</b> information now displays a sorted list of users. If a user is logged in more than once, ContactManager appends the number of sessions to the user name in parentheses. For example: <p>psmith(2),bsmith</p> The intent is to preserve all information that is currently available, while providing a view that is easier to scan because of the sorting and the non-repetition of names.
Persistence	
PL-17656	Guidewire has introduced a new class <code>com.guidewire.external.Type</code> to use for Java API methods that require an instance of <code>com.guidewire.external.Type</code> . <p>You can obtain an instance of this class using the <code>com.guidewire.external.Type#of</code> method by passing in the class representing the desired type. Methods that require an instance of <code>com.guidewire.external.Type</code> need to document what kinds of types are expected.</p>
Plugins	

PL-17162	Guidewire has added interface <code>gw.plugin.SharedBundlePlugin</code> . Any plugin that implements this interface shares the current bundle rather than creating a new one. For example, any plugins that implements <code>IPreUpdateHandler</code> now needs to add <code>gw.plugin.SharedBundlePlugin</code> to their list of implemented interfaces.
Profiling	
PL-17533	Fixed an issue with the (Server Tools) <b>Web Profiler</b> in which the <b>Group Frames</b> result page did not show detail information. The page now shows the detail information.
Queries	
PL-17676 PL-17719	Guidewire has deprecated the <code>RawQuery</code> property. Instead, create a <code>gw.api.filters.StandardQueryFilter</code> and use its <code>filterQuery</code> method or the <code>IQueryBeanResult.addFilter</code> method to apply the filter to a query.
Rules Infrastructure	
PL-17893	Guidewire has added a new <b>(Server Tools) Info Page</b> that shows the safe persisting order of the Preupdate rules. This page lists the order in which <code>ContactManager</code> runs the Preupdate rules for their root entities.
Security	
PL-10126	<p>Guidewire has added a new <code>Credentials</code> plugin to the base configuration. The <code>CredentialsPlugin</code> is similar to the <code>DBAuthenticationPlugin</code> and consists of the following three components:</p> <ul style="list-style-type: none"><li>Interface <code>CredentialsPlugin</code> with the following sole method:<ul style="list-style-type: none"><li><code>public UsernamePasswordPairBase retrieveUsernameAndPassword(String key)</code></li></ul></li><li>Class <code>CredentialsUtil</code> with the following sole static method:<ul style="list-style-type: none"><li><code>public static UsernamePasswordPairBase getCredentialsFromPlugin(String key)</code></li></ul></li><li>Class <code>CredentialsPlugin.gs</code> provides a default implementation of the plugin in the base configuration.</li></ul> <p>Code can call the method <code>CredentialsUtil.getCredentialsFromPlugin</code>, passing in a key to specify for which application the credentials are sought. The static method returns the username/password pair for the desired key. The default implementation currently reads the username/password pair from a <code>Credentials.xml</code> file as follows:</p> <ul style="list-style-type: none"><li>The code encounters the token <code>\${username}</code> as the value of the username value or the token <code>\${password}</code> as the value of the password value in the properties file. In this case, the code uses the <code>Credentials</code> plugin to obtain the real username and password.</li><li>The <code>Credentials</code> plugin is not enabled or these tokens are not present. In this case, the code simply uses the values that are present in the properties file for backwards compatibility.</li></ul> <p>However, you can substitute any mechanism for retrieving username/password pairs in place of the default mechanism. This can include retrieving username/password pairs from a directory service or from encrypted fields in a database.</p>
Studio IDE - Debugger	
PL-17929	Fixed an issue that caused a Null Pointer Exception, causing the interface to freeze. This issue occurred under specific circumstances after connecting Studio to the application server or starting a debugging session on an already-running server.
PL-18106	<p>Fixed an issue in which some actions in Studio, such as setting breakpoints, would block for a short while waiting for a server response, but would not provide any messages.</p> <p>Guidewire now shows a progress dialog to indicate the delay.</p>
PL-18110	<p>Previously, if the connection between Studio and the application server was lost for some reason, such as terminating the server session, Studio did not terminate the debugging session. Therefore, after server restart, it was possible for a user to believe that debugging was still ongoing, even though, in reality, the debugger needed to be reset.</p> <p>The new behavior is that Studio properly terminates the debugging session if it loses connection to the server, and then Studio notifies the user. The user must manually reactivate the debugger after connection to the application server is restored.</p>
PL-18112	Previously, a web service error while activating the Server debugger could cause the activation to fail, but Studio could show the debugger as active, even it was not. Studio now notifies you of the failure and properly puts the debugger into an inactive state.

Guidewire has also put into place additional recovery mechanisms from certain types of web services errors, making it less likely for the debugger connection to fail in the first place.

PL-18116	Fixed an issue that caused Studio to hang if you were debugging the server and the Gosu Tester debugger was also active. It also occurred if you has an active Gosu Tester debug session, then opened the Guidewire Studio connection dialog and immediately canceled. In that case Studio terminated the Gosu Tester debug session as well.
PL-18275	Fixed a problem with the remote tester debugger hanging if you disconnected it from the server during an active debug session.
PL-18298	Fixed a problem that could cause the Studio tester window to freeze up. The window could freeze if the user was using the tester debugger while connected to a remote server and modified some code in Studio and saved it.

#### Studio IDE - Other

PL-16798	Modified the Rule Condition parser to accept a statement list, instead of a simple expression only. However, the statement list must contain a return statement. For example: <pre>uses java.util.HashSet uses gw.lang.reflect.IType var o = new HashSet&lt;IType&gt;() {A, B, C, ...} return o.contains(typeof(...))</pre>
PL-17386	Added the ability to inspect inner class values in Gosu while debugging in Studio.
PL-17606	Fixed an issue in which Studio incorrectly handled creating a View entity extension.
PL-18107	Studio now shows a broken server connection state (icon) if it loses a server connection, for example, if the user terminates the server process. <ul style="list-style-type: none"> <li>It detects lost connections very quickly if debugging as there is immediate feedback from the debug API.</li> <li>It detects lost connections more slowly (but, with not more than 1 minute of delay), if not debugging, as Studio queries the server for status at one minute intervals.</li> <li>If Studio detects the server has restarted, Studio reconnects automatically.</li> </ul>
PL-18109	Fixed a problem that could occur if Studio was connected to a server that was not responding for a period of time. Multiple threads would pile up in the Studio process waiting for the server to respond. Guidewire has improved how it handles restoring the server connection with Studio. It now properly handles reconnecting to a server that was previously connected, but became unavailable for a period of time, and then became available again.
PL-18198	Fixed an issue in which that could occur if you set breakpoints in a Studio session and shut down and restarted Studio. Studio would lose the breakpoints between sessions if you activated the Server debugger after starting the application server, then shut down and restarted Studio.
PL-18227	Fixed an issue in which ContactManager stored the initial value of a newly created script parameter in an incorrect location, thus invalidating the application installation checksum. ContactManager did correctly store subsequent entries in the correct location.

#### Studio IDE - PCF Editor

PL-13845	Guidewire has modified the PCF <b>New Folder</b> dialog so that it is no longer possible to create a folder that differs in case only with an existing folder.
PL-11489	Guidewire has added a new PCF Verification option <b>Tools→Options→Verification Options</b> . Use it to enable or disable whether ContactManager limits the number of included sections in second-pass PCF verification. Second-pass verification can cause performance issues with combinations of modal PCF files.
PL-17576	Fixed an issue that caused Studio to throw an <code>AssertionError</code> while attempting to load a file, such as a PCF file, into Studio. If the file was modified outside Studio and a stale version of that file was already open in Studio, you would see this error on return to Studio. Studio now saves an open file properly on losing focus, and the error does not occur.



Before this fix, this error could happen if:

- You opened a file in Studio and shifted focus to another application such as IntelliJ. You then modified and saved the file and shifted focus back to Studio.
- You opened a file in Studio and shifted focus away from Studio. Then, through a source control management system, you reloaded a modified version of the file into the local file system and shifted focus back to Studio.

In each case, there was a mismatch between the file stored locally and the version of the file contained in Studio memory. The mismatch happened because Studio did not save the open file properly on losing focus.

Utilities	
PL-17920	Guidewire has removed the following JAR files from the application build: <ul style="list-style-type: none"> <li>• activation.jar</li> <li>• activation-1.1.1.jar</li> </ul>
PL-17922	Guidewire has removed the following JAR files from the Guidewire application builds as the standard Java EE install contains these files: <ul style="list-style-type: none"> <li>• mailapi.jar</li> <li>• smtp.jar</li> </ul>
Web - Other	
PL-17861	Added a disablePostOnEnter attribute to the Input Group widget.
Web - UI/Runtime	
PL-18233	Fixed an issue that failed to highlight (in yellow) a problematic RadioButtonCell field even if the data value for the RadioButtonCell contained an error.
Web Services - WSI	
PL-12215	When you place a WSDL from a web service publisher anywhere in the Gosu class path, Guidewire generates Gosu types from the WSDL elements. A problem in which these generated Gosu types could not be used successfully has been fixed.
PL-16577	To authenticate using HTTP authentication on WebLogic, add the following inside the security-configuration tag of WebLogic's config.xml: <pre>&lt;enforce-valid-basic-auth-credentials&gt;false&lt;/enforce-valid-basic-auth-credentials&gt;</pre>
PL-17412	Guidewire now provides a new class, WsiRequestLocal, for use with webservice request-based data storage. You can use this class to communicate values between various parts of a webservice implementation in the scope of a single request.
PL-17568	Fixed an XML parse exception (gw.xml.XmlException) that occurred if you invoked a web service that used a schema with multiple imports
PL-17601	Restored the functionality to extend an interface by a Gosu type. The WSDL was incorrectly overriding the final methods of the Object class (wait, notify, and similar methods). Eventually, when these WSDLs were transformed into Java classes, the compilation of the Java class failed because the final methods on Object could not be overridden. The fix was to not override these methods in WSDL generation.
PL-17902	Guidewire has added a check for the existence of the generated WSDL directories.
Workflow	
PL-9668	Added new configuration parameter WorkflowLogDebug, which takes a Boolean value: <ul style="list-style-type: none"> <li>• If set to true, ContactManager outputs the ordinary verbose system workflow log messages from the Guidewire server to the workflow log.</li> <li>• If set to false, ContactManager does not output any of the ordinary system messages.</li> </ul> <p>The setting of this parameter does not have any effect on calls to log workflow messages made by customers. Therefore, all customer log messages are output. If customers are experiencing too many workflow messages being written to the ab_workflowlog table, Guidewire recommends that you set this parameter to false.</p>



PL-17556	It is now possible to run the following writer batch processes from either the (Server Tools) <b>Batch Process Info</b> or (Server Tools) <b>Work Queue Info</b> pages: <ul style="list-style-type: none"> <li>• Activity Escalation</li> <li>• Group Exception</li> <li>• User Exception</li> </ul>
XML	
PL-17275	The Gosu tester in Studio now properly handles control characters in program output when connected to a running server.
PL-17934	Fixed an issue in which the XML subsystem did not parse complex types correctly if the complex types within XSD files made circular reference each other.

## Known issues and limitations for ContactManager 7.0.1

This section describes known issues with this release of Guidewire ContactManager.

- “ContactManager known issues” on page 457
- “Studio/Platform known issues” on page 457

**Note:** For maintenance releases, Guidewire often defers fixing configuration issues if they require merging files during the upgrade. Workarounds to many of these issues are listed in the following sections. The goal of this policy is to make upgrades as straightforward as possible.

### ContactManager known issues

#### Customizing contact relationship handling to use messaging results in two relationship records being created (CTC-396)

**Issue** – Changing a contact relationship updates two contact records. However, ContactManager stores the representation of the relationship in one `ContactContact` entity. ContactManager does not check to make sure the relationship does not already exist. Therefore ContactManager can create multiple, redundant `ContactContact` entities for the same relationship if it receives messages changing the relationship between two contacts.

**Workaround** – In the base configurations of d ContactManager, this behavior is not a problem. ClaimCenter makes a direct web service call to ContactManager to update contact relationships and does not use event messaging for related contact updates.

If you customize ClaimCenter and ContactManager to use event messaging to update related contacts between ClaimCenter and ContactManager, your customization must account for the extra message. Updating one relationship changes two contacts, and therefore ClaimCenter generates two messages and sends them to ContactManager. Your customization must ensure that ContactManager creates only one `ContactContact` entity in this case.

### Studio/Platform known issues

#### Issues with Internet Explorer 9

**Issue** – If you are using the Internet Explorer 9 browser, it is possible to see issues such as screen flickering or an incorrect tab order for fields. According to public reports, Internet Explorer 9 exhibits these and other issues with a variety of web sites and web applications.

**Workaround** – Because this is the behavior of the Internet Explorer 9 rendering engine, Guidewire cannot address these issues. However, there are reports of an Internet Explorer 9 workaround that reduces these issues. Change the new Internet Explorer 9 **Accelerated Graphics** settings on the **Advanced** tab of the **Internet Options** dialog.

#### First time you click the arrow of the typekey input, the drop-down menu does not open (PL-10134)

**Issue** – The drop-down menu does not open on the first click of the arrow on a typekey input. Instead, the help text opens.

**Workaround** – Turn off help text on focus by setting `InputHlepTextOnFocus` to `false` in the `config.xml` file. After you do that, the help text shows only if you mouse over the input, and it does not interfere with opening a drop-down menu.

#### XML API upgrade feature missing from documentation (PL-10257)

**Issue** – The *Integration Guide* describes a new set of XML APIs based on the `XmlElement` class. (Legacy APIs are based on the `XMLNode` class.) You can continue to use the legacy APIs. However, the *Integration Guide* omits mentioning an additional upgrade-specific feature.

**Workaround** – For backwards compatibility only, you can import an XML schema into the Gosu type system using the legacy XML system by following these instructions:

1. Copy:

```
ContactManager/modules/pl/config/registry/compatibility-xsd.xml
```

To:

```
ContactManager/modules/configuration/config/registry/compatibility-xsd.xml
```

2. Add an entry for your schema. Set the value of the `namespace` attribute to the Gosu package name of the schema. For example, if the schema is in the package location `my.package` and is called `myschema.xsd`, set the value of `namespace` to `my.package.myschema`.

#### ListDetailPanel throws exception (PL-10316)

**Issue** – It is possible for `ContactManager` to throw an exception if the user cancels out of a `ListDetailPanel` widget if `StartInEditMode` is also `True`.

**Workaround** – Set `StartInEditMode` to `False` for the screen that contains the `ListDetailPanel`. As a consequence, the user must click **Edit** to modify that screen.

#### Countries configured in zone-config.xml still generate a warning during regen-dictionary even when zone data is loaded for all these countries (PL-11947)

**Issue** – Countries configured in `zone-config.xml` still generate a warning during regen-dictionary even when zone data is loaded for all of these countries.

**Workaround** – Warning message is created in error and can safely be ignored.

#### There is a length limitation on entity localization table names (PL-13360)

**Issue** – There is a length limitation on entity localization table names.

**Workaround** – Ensure that the `localization tableName` property specified in the entity extension file is less than 16 characters. If the localization table name exceeds the maximum length, the error message indicates that 18 characters are allowed. However, the error message does not account for two additional characters added by the application.

#### GX models that reference virtual fields and enhancements throw null pointers if null (PL-13560)

**Issue** – The GX models that reference virtual fields and enhancements throw null pointers when these fields and enhancements are null.

**Workaround** – Include null checks and error handling to prevent referenced virtual fields or enhancements that are null from causing null pointer exceptions.

#### Sending email with file attachment with unicode filename is not correctly handed over to the mail server (PL-13582)

**Issue** – An email with a file attachment that has a unicode file name is not sent to the mail server correctly.

**Workaround** – Use Latin characters for file names on attached files.

### JavaToolkit.gs has incorrectly hard-coded memory, which results in failed regen-java-api Ant task (PL-13663)

**Issue** – The JavaToolkit.gs file has hard-coded memory, which can result in failed regen-java-api Ant tasks.

**Workaround** – Increase the size of the maximum heap setting on line 161 of JavaToolkit.gs in the Ant module. The default value is 512.

### Problem with regen-java-api command and JAR files (PL-16351)

**Issue** – If you run the ContactManager/bin/gwab regen-java-api command, ContactManager creates a ContactManager/java-api/lib directory and puts JAR files for Java development in this directory. However, ContactManager does not always perform this task correctly, resulting in some of the generated JAR files' not being copied to this directory.

**Workaround** – For missing library files, copy or import into your Java development environment the missing library files from the following directory:

```
ContactManager/admin/lib
```

Copy them into the following directory:

```
ContactManager/java-api/lib
```

### Studio Rules do not use correct capitalization for root object's name (PL-10740)

**Issue** – Rule set root objects are not named with first letter lowercased.

**Workaround** – The rules engine issues a warning if the correct case for objects is not being used.

### User interface cannot handle starting multiple instances of a batch process (PL-12372)

**Issue** – The user interface cannot handle starting multiple instances of a batch process.

**Workaround** – To execute multiple instances of a batch process, start them from the command line. Also, to allow multiple instances to be run simultaneously, you must ensure that the BatchProcess.isExclusive() method returns false.

### Type system refresh after PCF page title change does not update corresponding menu label (PL-13057)

**Issue** – The type system refresh after a PCF page title change does not update the corresponding menu label.

**Workaround** – After updating a page title, restart the server to refresh menu labels and avoid null pointer exceptions due to stale references.

### US-Locations.txt file with the US geodata from GreatData has special characters that cause validation problems with United States Postal Service (USPS) data (PL-13384)

**Issue** – The US-Locations.txt file contains information that does not conform to United States Postal Service (USPS) standards for bulk mailings.

**Workaround** – The provided US-Locations.txt file is intended only for use in geocoding to identify addresses for a location. You can edit the US-Locations.txt file to conform to your particular address standards, and then import that version of the file instead.

### GX model generated XSD cannot be parsed by JAXB (PL-13598)

**Issue** – XSD generated by the GX model cannot be parsed by JAXB.

**Workaround** – Add JAXB annotation elements to the XSD to specify the necessary metadata, such as class names, to enable JAXB to generate the Java class files. Contact Guidewire Support for a sample XSD file that is annotated this way.

### Cannot make a field from a delegate into a localized column (PL-13761)

**Issue** – You cannot make a field from a delegate into a localized column.

**Workaround** – Move the column to be localized off the delegate and onto each of the implementing entities. Then, to make the column appear as though it exists on the delegate, define an enhancement property on the delegate that *delegates* to the appropriate column.

### Studio test functionality not working correctly (PL-15153)

**Issue** – If you run a test in the **Tests** folder in the Guidewire Studio™ **Resources** tree, you get the following exception:

```
Using Test Environment Delegate: com.guidewire.testharness.ConfigEnvTestEnvironmentDelegate
gw.internal.gosu.parser.RuntimeExceptionWithNoStackTrace: java.lang.ClassNotFoundException:
 qa.DothisTest
Caused by: java.lang.ClassNotFoundException: qa.DothisTest
 at gw.internal.gosu.parser.TypeLoaderAccess.getIntrinsicTypeByFullName(TypeLoaderAccess.java:522)
 at gw.internal.gosu.parser.TypeSystemImpl.getByFullName(TypeSystemImpl.java:139)
 at gw.lang.reflect.TypeSystem.getByFullName(TypeSystem.java:116)
 at com.guidewire.studio.junit.ui.RunTestCommand.buildTestSuite(RunTestCommand.java:109)
 ...
```

**Workaround** – Run the following command before you run a test:

```
.../bin gwab dev-deploy
```

### Renaming method or property throws ParseResultsException (PL-16633)

**Issue** – If you rename a property or method or you change a method signature, and a workflow references that property or method in a Gosu field, ContactManager throws `ParseResultsException`. This is the intended behavior.

**Workaround** – Restart the workflow engine. To do so:

1. Log into ContactManager using an administrative account.
2. Access **Internal Tools**→**Reload**.
3. Click **Reload Workflow Engine**.

### Deploying EAR File on WebSphere 7.0.0.15 Generates Error Message (PL-18613)

**Issue** – The following steps generate an error message if you attempt to deploy a Guidewire-generated EAR file to WebSphere 7.0.0.15.

1. Generate an EAR file using the following command:

```
gwab build-websphere-ear
```

2. Deploy the EAR file to WebSphere.

The deployment fails with an error message similar to the following:

```
com.ibm.websphere.management.application.client.AppDeploy
Exception: com.ibm.websphere.management.application.client.AppDeploymentException:
 ADMA0207E: EE 5 module ab.war in ear file contains unsupported xmi format bindings file.
```

**Workaround** – Add the following at the top of the `web.xml` file before attempting to generate the EAR file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd">
```