



Universal Routing 8.1

Reference Manual

Notice:

All changes or additions to URS

that would appear in this manual

are in online [Supplement](#).

The information contained herein is proprietary and confidential and cannot be disclosed or duplicated without the prior written consent of Genesys Telecommunications Laboratories, Inc.

Copyright © 2010-2016 Genesys Telecommunications Laboratories, Inc. All rights reserved.

About Genesys

Genesys is the world's leading provider of customer service and contact center software - with more than 4,000 customers in 80 countries. Drawing on its more than 20 years of customer service innovation and experience, Genesys is uniquely positioned to help companies bring their people, insights and customer channels together to effectively drive today's customer conversation. Genesys software directs more than 100 million interactions every day, maximizing the value of customer engagement and differentiating the experience by driving personalization and multi-channel customer service - and extending customer service across the enterprise to optimize processes and the performance of customer-facing employees. Visit www.genesys.com for more information.

Each product has its own documentation for online viewing at the Genesys Documentation website or on the Documentation Library DVD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Genesys and the Genesys logo are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other company names and logos may be trademarks or registered trademarks of their respective holders. © 2014 Genesys Telecommunications Laboratories, Inc. All rights reserved. The Crystal monospace font is used by permission of Software Renovation Corporation, www.SoftwareRenovation.com.

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Customer Care from Genesys

If you have purchased support directly from Genesys, please contact [Genesys Customer Care](#). Before contacting Customer Care, please refer to the [Genesys Care Program Guide](#) for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the [Genesys Licensing Guide](#).

Released by

Genesys Telecommunications Laboratories, Inc. www.genesys.com

Document Version: 81r_ref_04-2016_v8.1.401.01



Table of Contents

Preface	19
Customer Interaction Management	20
eServices	20
Intended Audience.....	21
Making Comments on This Document	21
Contacting Genesys Customer Care.....	21
Document Change History	22
Release 8.1.4.....	22
Release 8.1.3.....	23
Release 8.1.2.....	23
Release 8.1.1.....	25
Chapter 1	Routing Strategies.....	27
Routing Strategies Defined.....	27
Routing Strategy Types	28
Strategies Created in IRD.....	28
Planning Strategies	30
Limitations	30
Configuration Layer Limitations	30
IRD Limitations	31
Creating Strategies.....	32
URS Internal Queues and Performance	33
Testing Strategies	34
Reviewing the Strategy Design.....	34
Compiling and Checking Integrity	34
Check Integrity Mode	34
The Debugging Tool.....	36
Loading a Strategy	38
Monitoring Strategies.....	38
IRD Monitoring View	39
Trace View	40
Sharing Strategies	42
Sharing Strategies Within the Same Environment.....	43
Sharing Strategies With a Different Environment or Tenant	43

Locking Strategies and Subroutines	44
Exporting/Importing Objects Between Strategies	44
Exporting Strategies	46
Export to File Menu Command	47
Exporting Using Archive (*.zcf) Format.....	48
Importing (Unpackaging) a Strategy	50
Making Strategies Ready for Immediate Execution.....	50
Import Operation	50
Migrating Strategies and Other Objects	54
Solution Export View.....	54
Solution Import View.....	58
Modifying Strategies	59
Specifying a Working Directory.....	59
Loading an Edited Strategy	60
Saving Strategies	60
Strategy Elements	60
Saving the RBN File	61
Default Storage Location by Tenant.....	61
Using Save As to Rename Strategies	62
Chapter 2	
Interaction Routing Designer Objects	63
Reusable Objects	64
Strategies.....	65
Subroutines.....	65
Routing Rules	70
Attributes.....	70
Business Rules	71
Interaction Data	72
Schedules	72
Statistics	73
List Objects	73
Macros	74
Strategy-Building Objects	74
Toolbar Icons	76
Buttons Associated with Toolbar Icons	76
Alphabetical List of Strategy-Building Objects	81
Object Properties Dialog Boxes.....	85
Logical Expressions.....	87
Expression Builder.....	87
Skill Expressions.....	88
Building Expressions	90
Variables in Objects and Expressions	92
Variable Usage and Strategy Design	92

Configuring Variables.....	93
Assigning Values to Variables.....	94
Exporting and Importing Variables	95
Specifying Variables.....	96
The Interaction Design Window.....	98
Business Processes	98
Opening Interaction Design	99
Creating a New Business Process	99
Business Attributes.....	102
Using Business Attributes	103
Using Objects in Strategies	104
Strategy-Building Objects Organization.....	104
Data and Services	107
Database Wizard	108
External Service.....	115
Web Service.....	119
Miscellaneous Objects.....	120
Assign	120
Call Subroutine	121
Error Segmentation.....	123
Entry	126
Exit.....	127
Function	127
If.....	129
Macro	130
MultiAssign	140
MultiAttach	140
MultiFunction	146
Multimedia Objects	149
Business Processes	149
Interaction Design Objects.....	150
About Genesys Queues.....	150
URS Communication with Interaction Server	151
Strategy-Building Objects For Business Processes	153
Strategy-Linked Nodes	153
Summary of Multimedia Objects	156
Finishing a Routing Strategy.....	161
Acknowledgement	161
Attach Categories	178
Autoresponse.....	181
Chat Transcript	191
Classify	198
Create Interaction	205
Create Email Out	212

Create Notification	217
Create SMS	221
Distribute Custom Event	226
Find Interaction	228
Forward E-Mail	233
Identify Contact.....	237
Last Agent Routing	241
MultiScreen.....	241
Redirect E-Mail	250
Render Message Content.....	254
Reply E-mail From External Resource	258
Screen	263
Send E-Mail	269
Stop Interaction.....	274
Submit New Interaction.....	277
Update Contact.....	281
Special Note on the Result Tab in Multimedia Objects	283
Update Interaction	285
Update UCS Record	290
Outbound Objects.....	294
Add Record.....	295
Do Not Call	299
Processed	302
Reschedule	305
Update Record.....	307
Configuring Proactive Routing	310
Routing Objects	311
Creating Routing Rules for Routing Objects	312
Default	316
Force Routing	316
Load Balancing	317
Percentage	318
Queue Interaction	320
Route Interaction	322
Selection	326
Service Level	333
Statistics	346
Switch-to-Strategy	347
Workbin.....	348
Workforce.....	353
Error Codes for Routing Objects.....	357
Postrouting Following a Routing Object.....	357
Virtual Queues in Routing Objects.....	357
Statistical Objects (Target Types)	358

Skill Target in Routing Objects	363
Skill Targets and Skill Expressions	365
Using a Skill Expression to Exclude an Agent	366
Important Note on the Absence of a Skill	366
Busy Treatments in Routing Objects	366
Targets in Other Objects	373
Segmentation Objects	374
ANI	374
Business	375
Classify	377
Date	380
Day of Week	381
DNIS	381
Generic	383
Screen	384
Time	385
SMS Objects.....	386
Create SMS Out	386
Send SMS Out.....	390
Workflow and Resource Management Objects	392
Force Logout.....	393
Set Agent DND State	395
Set Agent Media State	398
Set Multimedia Strategy State	400
Voice Treatment Objects	404
Mandatory Versus Busy	405
Updating Wait Time Information for an IVR	405
Beginning a Busy Treatment.....	406
Voice Treatment Parameters	406
Switch API Settings	406
Voice Treatment Error Messages	407
The Voice Treatment Objects	407
Access Control	418
Access Control Example.....	420
Access Control and Sharing Strategies	421
Chapter 3	
Interaction Routing Designer Functions	423
Value Types	424
Type Conversion.....	424
Conversion Rules	426
Key-Value List.....	428
STRING (Target).....	428
STRING (Statistical Object)	429

DN Target Format	429
Quote Usage, Variables, and Function Parameters.....	429
Specifying Variables in Dialog Boxes.....	430
Virtual Queues and DNIS Information.....	430
Summary of All Functions.....	430
Specialized Functions.....	449
CallInfo Functions.....	450
ACDQ	451
ANI.....	451
Attach.....	451
BearerCapability	453
BusinessData.....	453
BusinessDataINT	454
CallID	454
CallType	454
CallUUID.....	455
CED	455
ConnID.....	455
DeleteAttachedData.....	456
Dest	456
DNIS	456
ExtensionAttach.....	456
ExtensionData	457
ExtensionUpdate	457
FirstHomeLocation.....	457
GetCurrentLeg	457
GetCurrentSwitch	458
GetCurrentTServer	458
GetCustomerSegment	459
GetMediaType	459
GetRoutingPoint	460
GetServiceObjective	460
GetServiceType	460
InformationDigits	461
InteractionData	461
InteractionDataINT	462
LATA	462
NPA.....	462
NPANXX	463
Orig	463
OtherTrunk.....	463
PACCCode	463
PACType	463
RequestType.....	464

SetHomeLocation	464
StateCode	465
ThisTrunk	465
UData	465
UDataINT	466
Update	467
UpdateBusinessData	468
UpdateInteractionData	469
Configuration Options Functions	469
ExcludeAgents	470
FindConfigObject	471
GetConfigOption	472
GetCurrentScript	473
GetObjectProperty	473
GetMediaTypeName	474
Router	474
SetCallOption	475
SetDNIS	475
SetDNISOverride	475
SetObjectProperty	476
SetTranslationOverride	476
UseActivityType	477
UseCustomAgentType	478
UseCustomDNTYPE	478
UseCustomPlaceType	478
UseDNTYPE	479
UseMediaType	479
Data Manipulation Functions	480
FindServiceObjective	480
ListGetDataCfg	482
ListLookup	483
ListLookupCfg	484
Date and Time Functions	485
Date	486
DateInZone	486
Day	486
DayInZone	487
GetUTC	487
IsSpecialDay	488
IsSpecialDayEx	488
Time	489
TimeDifference	489
TimeInZone	490
TimeStamp	490

UTCAdd	490
UTCFromString.....	491
UTCToString	492
Force Functions.....	492
Force.....	493
TRoute	493
List Manipulation Functions	494
GetIntegerKey.....	495
GetMaxSubList	495
GetMinSubList	495
GetStringKey	495
GetRawAttribute	496
KVList Functions.....	497
KVListFindSubList	497
KVListGetKey	497
KVListGetListValue	498
KVListGetSize.....	498
KVListGetStringValue	498
ListGetInteger	498
ListGetKey	499
ListGetSize	499
ListGetString	499
SetIntegerKey and SetStringKey	500
SetIntegerKey	500
SetStringKey	500
TargetState	500
Miscellaneous Functions	505
ActiveServerName	505
Alarm	506
AnswerCall.....	506
CheckAgentState	507
ClaimAgentsFromOCS	508
ClearTargets	510
ClearUpdateTrigger	511
CountSkillInGroupEx	511
CountTargetsByThreshold	514
CreateSkillGroup	514
Delay.....	515
ExpandGroup.....	515
ExpandWFActivity	517
ExtrouterError	518
ExtrouterStatus	518
GetLastErrorInfo	518
GetPriority	519

GetSkillInGroupEx	519
JumpToStrategy	522
JumpToTenant	523
MultiplyTargets	523
MultiSkill.....	526
OnCallAbandoned	527
OnRouteError	527
Print	528
Rand	529
ReleaseCall	529
ResetBusyTreatments	530
SelectTargetsByThreshold	530
SelectTargets	530
SendEvent	532
SendRequest	534
ServerStatus	544
SetDelayedAttach	545
SetInteractionAge	545
SetLastError.....	546
SetQueueLabel	548
SetRunTimeMode	548
SetUpdateTrigger.....	549
StrFormatTime	550
SuspendForEvent	550
TargetComponentSelected	551
TargetObjectSelected	552
TargetSelected	552
Timeout.....	553
UpdateScript	554
UseAgentState.....	554
UseAgentStatistics.....	555
VQSelected.....	555
Reporting Functions	556
Peg	557
PegValue.....	557
Statistical Functions.....	558
CallsDistributed.....	560
CallsEntered	560
CallsWaiting	561
ClearThresholds	562
DistributedPercentage	562
DistributedWaitingTime	563
GetAvgStatData	564
GetMaxStatData	564

GetMinStatData	565
InVQWaitTime.....	565
NotDistributedPercentage.....	566
NotDistributedWaitingTime	567
PositionInQueue	568
ResetStatAdjustment	569
RvqData.....	570
SData.....	572
SDataInTenant	574
SetStatAdjustment.....	574
SetStatUpdate	575
SetThresholdEx	577
UseCapacity	579
String Manipulation Functions	581
Bytes.....	581
Cat	582
StrAsciiBreak	582
StrAsciiTok	582
StrChar	583
StrGetChar.....	584
StrLen	584
StrNextTokInd	585
StrReplace	585
StrStr.....	585
StrSub.....	585
StrTargets	586
StrToLower.....	587
StrToUpper.....	588
Target Manipulation Functions.....	588
BlockDN	589
CCTExtractTargets	590
DeliverCall	591
DeliverToIVR	593
GetRemoteAccessCode	594
IncrementPriority.....	596
IncrementPriorityEx	596
KeepQueue	597
NMTEExtractTargets	598
Priority.....	602
PriorityLimits	603
PriorityTuning	603
RouteCall	607
Routed	607
SelectDN.....	608

SetTargetThreshold	612
SetVQPriority	613
SuspendForDN	614
SuspendForTreatmentEnd.....	615
TargetSelectionTuning	615
Translate	615
Retired Functions	616
CountSkillInGroup.....	616
Default	617
GetLastError	617
GetSkillInGroup	617
Goto	617
Increment.....	617
InvokeNirvanaMethod	617
OnError	617
SetThreshold	617
SetTreatmentMode	617
Skill	618

Chapter 4	Configuration Options	619
	Setting Options	620
	URS Options	620
	Summary of Options.....	621
	List of URS Options	622
	Location in Configuration Layer By Precedence.....	628
	URS Option Descriptions.....	629
	agent_att.....	629
	agent_reservation	629
	alternative_server	631
	automatic_attach	632
	backup_mode	632
	call_kpl_time	633
	call_monitoring.....	634
	call_tracking	634
	change_tenant.....	635
	check_call_legs	635
	compat_treatments	636
	count_calls.....	637
	cpu_emergency_level.....	637
	default_db_server	637
	default_destination.....	638
	default_object	638
	default_stat_server	639

emergency_verbose	639
environment	639
event_arrive	640
extrouter_dn	641
extrouter_timeout	641
function_compatibility	642
give_treatment	642
hanged_call_time	643
hot_backup_priority	643
hide_private_data	643
ignore_customer_id	644
inv_connid_errors	644
lds	645
management_port	645
max_cpu_objects_slice	646
max-submission rate	646
monitoring_time	646
null_value	647
on_primary_changed	647
on_route_error	648
on_router_activated	649
pickup_calls	650
prestrategy	651
pulse_time	652
reconnect_time	653
reduced	653
reg_attempts	654
reg_delay	655
report_reasons	655
report_statistics	658
report_targets	660
request_timeout	663
reservation_pulling_time	664
route_consult_call	664
run_time_mode	666
service_timeout	666
skip_targets	667
startup_verbose	667
strategy	668
targets_order	668
tattributes	669
Threshold	669
ThresholdDestination	669
ThresholdGroup	669

ThresholdSwitch	670
transfer_time	671
transfer_to_agent.....	671
transit_dn	671
transition_time	672
treatment_delay_time	672
unix_socket_path.....	673
unloaded_cdn	673
use_agent_att.....	674
use_agent_capacity	674
use_agentid	676
use_dn_type	677
use_extrouter.....	678
use_extrouting_type	679
use_ivr_info	680
use_parking_threshold	680
use_service_objective	681
use_translation	682
using	683
validate	683
verbose	684
verification_mode.....	686
verification_time	686
verification_time_agent.....	687
verification_time_dn.....	687
Web Services Options	687
http_port.....	688
http_log_file	688
http_log_size.....	689
log_buffering	689
log_remove_old_files.....	689
verbose	689
def_certificate	690
def_certificate_key	690
def_trusted_ca	690
soap_conn_idle_timeout.....	691
soap_retry_attempts	691
soap_retry_timeout	691
http_conn_idle_timeout.....	691
wfm_polling_interval	692
Web API (URS-Behind) Options.....	692
http_port.....	692
soap_port.....	693
log_file	693

log_buffering	693
log_remove_old_files	694
IRD Options	695
bytecode	695
inactivity-timeout	695
Custom Server Options	697
async	697
buffer_size	697
frequency	698
hide_private_data	698
life_time	698
new_parsing	698
Routing-Related Interaction Server Options	699
Routing-Related Message Server Options	701
ADDP Options for Universal Routing Server	701
Retired Options	703
Chapter 5 Number Translation.....	705
Configuring the use_translation Option	705
Configuring Translation Tables	706
Selecting a Target Type	707
Selecting a Route Type	707
Parameters for Switch Access Codes	707
Configuring the Reason Source and Extension Source	709
Chapter 6 Automatically Attached Pegs	711
Explicit Pegs	711
Automatic Pegs	711
Chapter 7 Routing Statistics	715
About This Chapter	716
Predefined Statistics	716
Use of Predefined Statistics in IRD	717
List of Predefined Statistics	718
Load Balance Family of Statistics	725
Router Self-Awareness	726
URS and Stat Server	727
Stat Server and Statistics	728
Other Routing Issues	729
Resolving Routing Issues	731
User-Defined Statistics	732

	Creating a New Statistic	732
	Statistics Properties Dialog Box.....	734
	Pseudo Statistics	736
Chapter 8	Solution Reporting	741
	Reporting in a Typical Contact Center.....	741
	Solution Reporting	742
	CCPulse+.....	742
	CC Analyzer.....	744
	Interaction Concentrator	745
	Reporting Documentation.....	748
	Reporting Templates.....	749
	Reports From Interactive Insights.....	749
	Agent Reports.....	749
	Queue Reports	749
	Business Reports.....	750
	Interactive Insights Documentation.....	750
Appendix A	Log Events	751
	Management Layer Log Output Levels	752
	Logging During Normal Operation	752
	Logging During Irregular Operation	753
	Log Filtering	753
	Universal Routing Log Messages	753
	Changing Log Message Values	754
	URS Log Message List	754
Appendix B	IRD Web Service Object.....	771
	URS Options Affecting Web Service	772
	Web Service Object.....	772
	Sample Strategy	773
	General Tab	774
	Nested WSDL Access	777
	Security Tab.....	778
	HTTP Authentication Area	779
	TLS	782
	Soap Security Area.....	783
	Result Tab.....	786
	Function Object #1.....	787
	Function Object #2.....	788
	How the Web Service Object Works.....	790

Limitation of the Web Service Object.....	790
Another Web Service Example.....	791
Web Services Resources	796
Appendix C	
URS-Behind Solution	797
URS-Behind Solution Introduction.....	797
Message Formats.....	798
R-COMMAND MESSAGES	798
R-COMMAND-CONTROL MESSAGES	799
Supported Interfaces	800
Direct Connection to URS.....	800
SOAP/Plain HTTP Interface	803
SOAP Interface	803
Supported Methods	811
SOAP Methods	811
Plain HTTP Methods.....	813
Supplements	
Related Documentation Resources	821
Document Conventions	824
Index	
.....	827



Preface

Welcome to the *Universal Routing 8.1 Reference Manual*. This manual contains technical information on both Enterprise Routing and Network Routing (ISCC in T-Server documentation), which together comprise Universal Routing. In brief, you will find information about the following: routing strategies, objects, functions, options, and statistics.

This document is valid only for the 8.1 release(s) of this product.

Note: For versions of this document created for other releases of this product, please visit the Genesys Documentation website, or request the Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesys.com.

This preface contains the following sections:

- [Customer Interaction Management, page 20](#)
- [Intended Audience, page 21](#)
- [Making Comments on This Document, page 21](#)
- [Contacting Genesys Customer Care, page 21](#)
- [Document Change History, page 22](#)

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on [page 821](#).

Customer Interaction Management

Universal Routing enables you to design sophisticated strategies for handling both voice and non-voice interactions and for directing them to an appropriate target. Universal Routing also provides data required to report on interaction handling in your enterprise.

Universal Routing, which comprises Enterprise Routing and Network Routing, is one part of the Genesys Customer Interaction Management (CIM) Platform.

The CIM Platform consists of the following:

- Genesys Universal Routing
- Genesys Reporting and Analytics
- Genesys eServices
- Genesys Management Framework

Each has its own documentation set.

eServices

Genesys eServices is the core of a series of components that work together to handle interactions from disparate media-based devices. It allows you to centralize your handling of the various channels that customers use to reach your contact center. The core functionality provided by eServices must operate with at least one of the following media channels:

- Genesys E-mail. This channel has an optional enhancement: Genesys Content Analyzer, which uses natural language technology to provide automated classification of incoming e-mail.
- Genesys Web Media (chat).
- Genesys Open Media, which allows you to add customized support for other media. For more information see the documents for the Genesys Developer Program 8.0, particularly those dealing with the Media Interaction SDK.

Note: Universal Routing 8.1 can work in a pure voice environment or with the eServices software components, which allow for the additional routing of non-voice interactions.

Intended Audience

This document is primarily intended for users involved in developing and setting up a routing solution with Interaction Routing Designer (IRD), including administrators and strategy designers. It has been written with the assumption that you have a basic understanding of:

- Computer-telephony integration concepts, processes, terminology, and applications.
- Network design and operation.
- Your own network configurations.

You should also be familiar with Framework architecture and functions and Genesys eServices (if installed).

Making Comments on This Document

If you especially like or dislike anything about this document, please feel free to e-mail your comments to Techpubs.webadmin@genesys.com.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this document. Please limit your comments to the information in this document only and to the way in which the information is presented. Speak to Genesys Customer Care if you have suggestions about the product itself.

When you send us comments, you grant Genesys a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

Contacting Genesys Customer Care

If you have purchased support directly from Genesys, please contact [Genesys Customer Care](#).

Before contacting Customer Care, please refer to the [Genesys Care Program Guide](#) for complete contact information and procedures.

Document Change History

This section lists topics that are new or that have changed significantly since the first release of this document.

Doc Change: Table 34 (page 202) paramaters updated. Added link to online [Supplement](#).

Release 8.1.4

Updated Chapters include:

- Chapter 1, “Routing Strategies” on [page 27](#):
 - “Migrating Strategies and Other Objects” on [page 54](#).
- Chapter 2, “[Interaction Routing Designer Objects](#)”:
 - “Creating a New Business Process” on [page 99](#), Step 2.
 - “Simple Versus Complex Macros” on [page 136](#).
 - “Interaction Priority and Service Level Routing” on [page 345](#).
- Chapter 3, “[Interaction Routing Designer Functions](#)”:
 - FindConfigObject moved to “Configuration Options Functions,” [page 471](#).
 - “GetObjectProperty,” [page 473](#) and “SetObjectProperty,” [page 476](#).
 - “CreateSkillGroup,” [page 514](#).
 - “Retired Functions” [page 616](#).
- Chapter 4, “[Configuration Options](#)”:
 - Descriptions for `tenant` and `DN`, “Setting Options” on [page 620](#).
 - Descriptions for options, “URS Option Descriptions” on [page 629](#):
 - Support of multiple and prioritized backup servers, `alternative_server`, `backup_mode`, `hot_backup_priority`
 - Manage and control CPU consumption: `cpu_emergency_level`, `emergency_verbose`, `startup_verbose`
 - Agent statistics options: `agent_att`, `use_agent_att`
 - “Retired Options” on [page 703](#).
- Chapter 7. “[Routing Statistics](#)”:
 - RStatRoundRobin [page 725](#).
- Appendix A: “[Log Events](#)”:
 - #442 in the “URS Log Message List” on [page 754](#).
- Appendix B: “[IRD Web Service Object](#)”:
 - Specifying authentication credentials, setting TLS parameters on the `web service properties`> Security tab: Certificate, Certificate Key, Trusted CA, [page 782](#)
- Appendix C: “[URS-Behind Solution](#)” on [page 797](#):
 - Running strategies for the SOAP method, [page 811](#) and the Plain HTTP method, [page 813](#).
 - URS REST web API method `report`, [page 816](#).

Release 8.1.3

- This release of URS supports Genesys SIP cluster solution for enterprise telephony, which is currently under restricted release. To learn more about Genesys SIP cluster solution, please contact your Genesys representative.
- URS now supports log filtering by provisioning a tag, where tag will cause KVList value to be prefixed by '<#' and post fixed by '#>. Users who want to have sensitive data in logs for their own purpose would be able to fetch these utilizing automated tools. See “Log Filtering” on [page 753](#).
- URS enhances security by supporting FIPS (Federal Information Processing Standards) compliant Transport Layer Security (TLS).
- IRD now provides the ability to display any meta-text associated with a successful external authentication through RADIUS Server upon user login.
- Chapter 2, “[Interaction Routing Designer Objects](#)”:
 - URS can now associate a Business Process with Accounts. See “[Associating a Business Process with Accounts](#)” on [page 101](#).
- Chapter 3, “[Interaction Routing Designer Functions](#)”:
 - The `ExtensionData` function has been updated. See “[ExtensionData](#)” on [page 457](#).
 - The `GetLastErrorInfo` function has been updated. See “[GetLastErrorInfo](#)” on [page 518](#).
- Chapter 4, “[Configuration Options](#)”:
 - The configuration option `run_time_mode` was added on [page 666](#)
 - Support for the Reset Password configuration attribute is introduced. See “[inactivity-timeout](#)” on [page 695](#)
- Chapter 7. “[Routing Statistics](#)”:
 - Media Statistic was added to the Statistics Properties Dialog Box. See “[Media Statistic](#)” on [page 735](#).

Release 8.1.2

- URS 8.1.2 continues to support Genesys Orchestration Server (ORS) 8.x and above. This combination of URS and ORS is known as Orchestration Platform.
- URS now supports distributed agent reservation, not only for voice interactions but also for Multimedia interactions too.
- URS now provides more robust logic to restore connections to Configuration Server, specifically in the case where the environment has redundant Configuration Servers.
- URS now places in the log the corresponding Log Events message (one of following) when it changes its run time mode (primary/backup):

- 4560|STANDARD|GCTI_REDUNDANCY_WARM_STANDBY_BACKUP_ACTIVATED|Warm Standby (backup) mode activated
- 4561|STANDARD|GCTI_REDUNDANCY_HOT_STANDBY_BACKUP_ACTIVATED|Hot Standby (backup) mode activated
- 4562|STANDARD|GCTI_REDUNDANCY_WARM_STANDBY_PRIMARY_ACTIVATED|Warm standby (Primary) mode activated
- 4563|STANDARD|GCTI_REDUNDANCY_HOT_STANDBY_PRIMARY_ACTIVATED|Hot Standby (Primary) mode activated
- Chapter 1, “[Routing Strategies](#)”:
 - IRD now optionally does not allow access to a subroutine if it is used by another strategy or subroutine. See “Saving Strategies” on [page 60](#).
- Chapter 2, “[Interaction Routing Designer Objects](#)”:
 - Additional parameters grid were added to Classify and Screen objects. See “Classify” on [page 198](#) and “MultiScreen” on [page 241](#).
 - IRD now verifies for a wait-time value on a target selection block. See “Target Selection Tab” on [page 325](#) and “Target Selection Tab” on [page 330](#).
 - Custom Routing functionality now allows automatic splitting of code. This splits target selection object into a set of functions (`SelectDN`, `SuspendForDN`, `RouteCall`) and allows to insert some additional actions between them. See “Route Interaction” on [page 322](#) and “Selection” on [page 326](#).
 - Delete unused target objects. See “Deleting Unused Target Objects” on [page 363](#)
 - Variables are resolved at the moment busy treatments are queued. See “Busy Treatments in Routing Objects” on [page 366](#)
- Chapter 3, “[Interaction Routing Designer Functions](#)”:
 - The function “CountTargetsByThreshold” on [page 514](#) has been added.
 - Date Time stamp function added. See “StrFormatTime” on [page 550](#).
- Chapter 4, “[Configuration Options](#)”:
 - URS now provides options to propagate or mask VQ event distribution. In a distributed environment this allows users to limit the number events propagated across Routers. See “transit_dn” on [page 671](#).
 - URS option `verification_time_dn` description updated. See “`verification_time_dn`” on [page 687](#).
- Chapter 7. “[Routing Statistics](#)”:
 - Router introduces a new pseudo statistics `RStatAgentsReadyMedia` and `RStatAgentsReadyvoice`. These Statistics are utilized to calculate the number of ready Agents for a specific media type, when Agents/Agent Groups/Places/Place groups/skill expressions are used as a target. See “`RStatAgentsReadyMedia`” on [page 724](#) and “`RStatAgentsReadyvoice`” on [page 724](#).

- URS statistics definition has been extended. See “Statistics Properties Dialog Box” on [page 734](#).
- Appendix B. “IRD Web Service Object”:
 - URS now allows HTML requests without any SOAP tags, which allows customers to send secure mail HTML requests from URS, when their environment does not support SOAP due to security reasons. See “General Tab” on [page 774](#)
 - WSDL sample for HTTP interface. See “How the Web Service Object Works” on [page 790](#)
- Appendix C. “URS-Behind Solution”:
 - The detailed diagnostic of internal virtual agent groups are now available upon running different types of queries.

Release 8.1.1

- Chapter 1, “Routing Strategies”:
 - A limitation was added to the section, “IRD Limitations” on [page 31](#).
- Chapter 2, “Interaction Routing Designer Objects”:
 - The note under the sub-heading “Schedules” on [page 72](#), has some additional information.
 - Two new sections have been added:
 - “CheckBusinessRule Macro” on [page 138](#)
 - “Support for Virtual Queues Reporting” on [page 358](#)
 - “Dynamic Busy Treatment at Selection Block” on [page 372](#)
 - Two rows in Table 80, “Submit New Interaction, General Tab,” on [page 279](#) were updated, “Parameters” and “Use User Data”
- Chapter 3, “Interaction Routing Designer Functions”
 - Table 130, “Interaction Routing Designer Functions,” on [page 430](#) has been updated to add new functions.
 - A new paragraph was added to section, “GetConfigOption” on [page 472](#) to further describe the option name.
 - The following new sections have been added to describe options:
 - “GetCurrentScript” on [page 473](#)
 - “GetIntegerKey” on [page 495](#)
 - “KVLListFindSubList” on [page 497](#)
 - “TargetState” on [page 500](#).
 - “CountSkillInGroupEx” on [page 511](#)
 - “GetSkillInGroupEx” on [page 519](#)
 - “MultiplyTargets” on [page 523](#)
 - “SDataInTenant” on [page 574](#)
 - Two new graphics have been added in the section, “MultiplyTargets” on [page 523](#). See [Figures 226](#) and [227](#).

- The section, “SuspendForEvent” on [page 550](#) has additional information.
- Chapter 4, “[Configuration Options](#)”
 - Table 136, “URS Options,” on [page 622](#) has been updated to add, remove, and revise options.
 - A note has been added to the section, “call_tracking” on [page 634](#).
 - The following sections have been updated with additional information to describe these options:
 - “hanged_call_time” on [page 643](#)
 - “use_extrouting_type” on [page 679](#)
 - A new subsection, “Restoring the Connection to Configuration Server” on [page 702](#) has been added.
 - The section, “Retired Options” on [page 703](#) has been updated to include any options are retired in this release.
- Chapter 7. “[Routing Statistics](#)”
 - A new subsection, “Failure to Open a Statistic” on [page 731](#) has been added.



Chapter

1

Routing Strategies

This chapter provides a high-level description of routing strategies in preparation for the remaining chapters. It contains the following sections:

- [Routing Strategies Defined, page 27](#)
- [Planning Strategies, page 30](#)
- [Limitations, page 30](#)
- [Creating Strategies, page 32](#)
- [Testing Strategies, page 34](#)
- [Loading a Strategy, page 38](#)
- [Monitoring Strategies, page 38](#)
- [Sharing Strategies, page 42](#)
- [Exporting/Importing Objects Between Strategies, page 44](#)
- [Exporting Strategies, page 46](#)
- [Importing \(Unpackaging\) a Strategy, page 50](#)
- [Migrating Strategies and Other Objects, page 54](#)
- [Modifying Strategies, page 59](#)
- [Saving Strategies, page 60](#)

Routing Strategies Defined

Routing strategies (strategies) are sets of simple or complex instructions that tell Universal Routing Server (URS) how to handle an interaction. The strategies can take many conditions into account, for example,

- The time and date of the interaction
- Identifying information associated with the interaction
- Agent availability
- sets
- Words or word patterns in text-based interactions

In addition, the strategies can instruct URS to apply various kinds of treatments to voice interactions.

Routing Strategy Types

In Universal Routing 8.1, there are two ways to create routing strategies:

1. Using Genesys Composer, which lets you create SCXML-based strategies. For more information on these types of strategies, see the chapter on SCXML strategy support in the *Universal Routing 8.1 Deployment Guide*.
2. Using Interaction Routing Designer (IRD), which creates strategies in the Genesys IRL routing language. [Figure 3](#) shows the IRD main window

With either method of creation, Universal Routing Server (URS or “Router”) executes the routing strategy instructions.

Strategies Created in IRD

You build and test IRL-based strategies in the Interaction Routing Designer (IRD) *Routing Design* window. Various types of interaction handling are represented by objects that you connect together to create a strategy. Icons on the toolbar at the top of the *Routing Design* window (see [Figure 1](#)) represent the following categories of objects:

- Data and Services (see [page 77](#))
- Multimedia (see [page 79](#))
- Miscellaneous (see [page 79](#))
- Outbound (see [page 80](#))
- Routing (see [page 78](#))
- Segmentation (see [page 78](#))
- Voice Treatment (see [page 76](#))

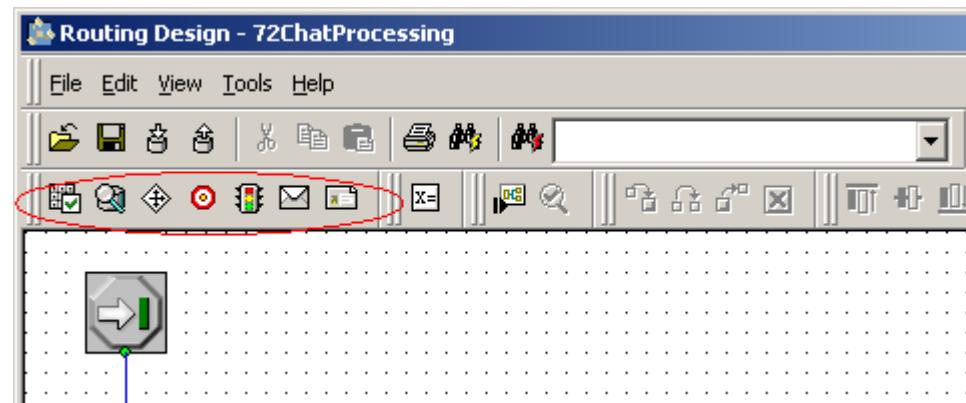


Figure 1: Routing Design Window Object Toolbar

Clicking the icon for each object category opens a sub-toolbar that contains additional buttons. For example, click the Segmentation icon to view buttons that represent the different types of Segmentation objects (see [Figure 2](#)).



Figure 2: Segmentation Toolbar

You use these buttons to place objects in the *Routing Design* window workspace, in which you build strategies (and subroutines). [Figure 3](#) shows an example strategy that uses Entry, Segmentation, and Routing objects.

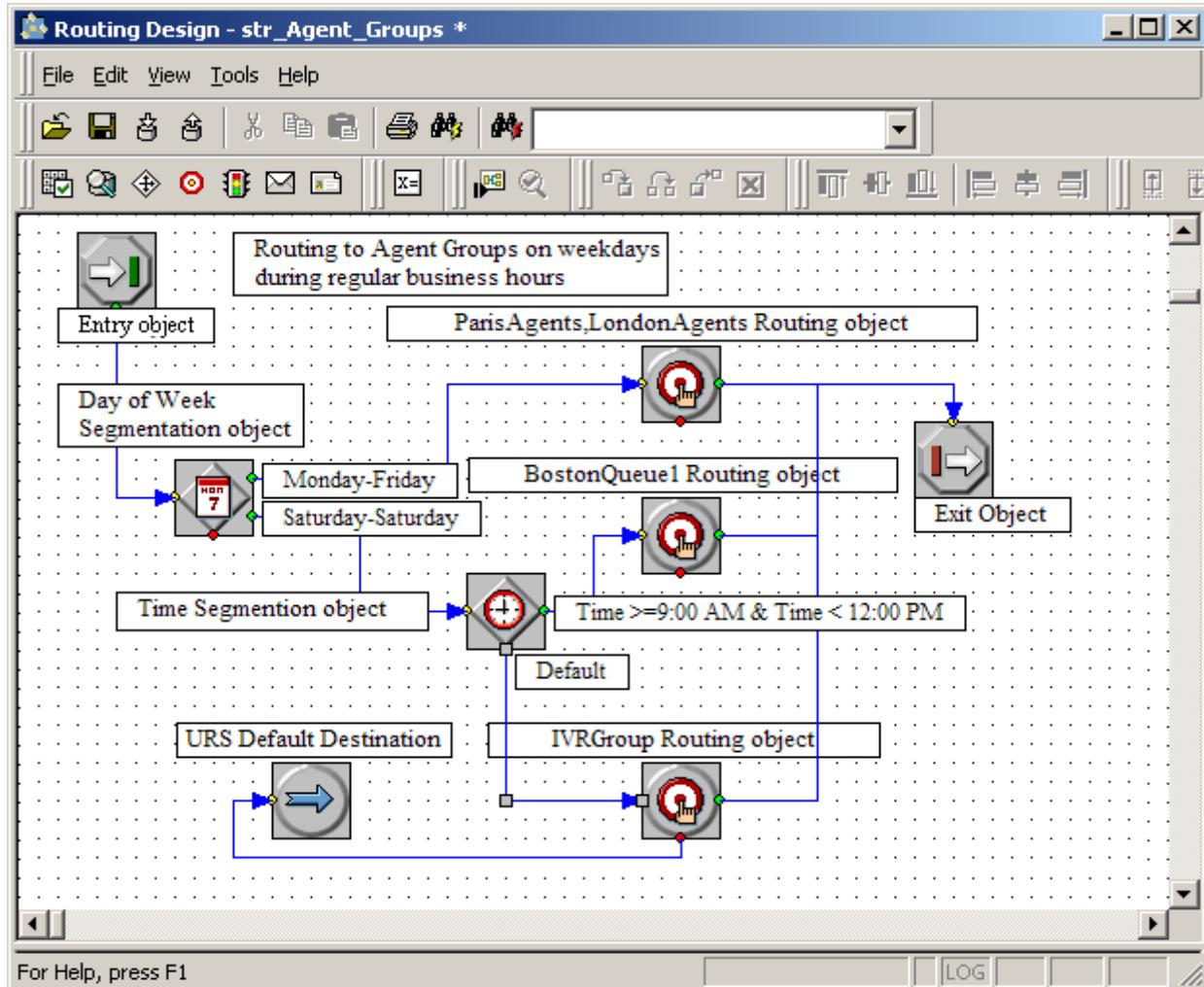


Figure 3: Example Strategy in the Routing Design Window

For more information about IRD objects, see Chapter 2, “Interaction Routing Designer Objects,” on [page 63](#) and the *Interaction Routing Designer 8.1 Help*. For sample strategies, see the *Universal Routing 7.6 (or later) Strategy Samples* document.

Planning Strategies

Before you can start developing strategies, you must do some business planning to help determine your routing objectives. Your routing objectives become the basis for designing strategies that meet your business needs.

In order to be translated into a routing strategy, routing objectives should be as specific as possible, and they also should be measurable in some way.

- For information about the types of routing that are supported, see the *Universal Routing 8.1 Deployment Guide*.
- For assistance in planning routing strategies—including considerations about how your contact center operates, the types of interactions you expect, and the sorts of reports you need—see the chapter on deployment planning in the *Universal Routing 8.1 Deployment Guide*. Also see the section on Universal Routing capabilities in Chapter 1 of that guide.

Limitations

When developing a strategy or a business process, familiarize yourself with the following limitations pertaining to Configuration Manager and IRD:

Warning! If you do not observe these limitations, your interactions might be routed to an incorrect destination, URS might unexpectedly shut down, or a requested operation might not be performed.

Configuration Layer Limitations

- While the Configuration Layer supports the full character set for use in object names, using certain characters can cause problems in the behavior of other Genesys applications.

The objects affected by this limitation that are used in Universal Routing include Tenant, Field, Format, Script (strategies and subroutines), Table Access, Tenant, and Transaction (routing rules, interaction data, attributes, business rules, and custom statistics).

- The maximum length of any Configuration Server object name is 254 bytes (a limitation that is automatically enforced by Configuration Server).
- When configuring Employee IDs, Login IDs, Place names, DN numbers, and Switch names that will be used to define routing targets, the maximum length is 63 bytes.
- When a URS name is added to the name of a T-Server that is connecting through a Load Distribution Server, the sum of the bytes cannot exceed 126 bytes.

- When defining agent states on the Annex or Options tab of a URS Application object (for example when preparing to use the UseAgentState function), the maximum number of user-defined agent states is 32.
- The names of skills, if they are used in a skill expression, cannot exceed 126 bytes.
- The priority of an interaction cannot exceed 1,000,000,000.

IRD Limitations

When creating or modifying strategies in IRD:

- Certain symbols, listed in [Figure 4](#), should not be used in object names. The names of all IRD reusable objects and data (strategies, routing rules, schedules, and so on) as well as Interaction Design window configuration objects (business processes, queues, and so on) are limited to alphanumeric characters and cannot begin with a space, an underscore, or a hyphen. Please note that although some IRD objects allow you to enter names containing special characters, Genesys strongly recommends not using them.



Figure 4: Invalid Symbols and Characters

- The underscore is the only character that you can safely use in the middle of the name of a strategy or other reusable object, service, method, or method parameter name of any strategy object that results in executing an external or internal service or procedure.

Warning! Using commas and square brackets as data separators can make IRD incorrectly interpret the entered values if those symbols are used in Treatment, Data and Services, Multimedia, and Outbound objects.

- When specifying busy treatment parameters, do not include pipes "|" and colons ":" in values.
- Skill expressions (see [page 88](#)) that are used by the Selection (see [page 326](#)), Service Level (see [page 333](#)), and Route Interaction (see [page 322](#)) Routing objects cannot exceed 100 elements (skill names, numbers, comparisons, and logical operands). That is, a skill expression should have no more than 25 constructions, such as English > 1. It is especially important to observe this limitation in the case of Service Level

routing rules, which use skills internally—that is, every skill criterion used in a Service Level routing rule generates 1–3 constructions in the form <Skill> > <Number>.

- The maximum size of an overall skill expression (as text) is 10239 bytes.
- The maximum length of a string using the format `placename@statservername.AP agentname@statservername.A` for any routing target is 253 bytes.
- The maximum length of a request to Custom Server or DB Server through XData (used for 5.x strategies) is 10239 bytes.
- When creating a key-value list that uses a string format (such as `aaa:5|gggg:123|...`), the maximum key length is 1000 bytes and the maximum length of one key-value pair is 4096 bytes.
- When using the Database Wizard object to query a database, the maximum size of a SQL statement created by the DB Wizard is 1010 bytes.
- When using the IRD statistical adjustment functions, the name of an object plus the name of the statistic cannot exceed 250 bytes.
- The name of any single target (including a skill group) cannot exceed 254 bytes.
- Integer variables can have a maximum value of approximately four billion, with a range of -2 billion to +2 billion. If a value number exceeds the maximum capacity of an integer variable, the number wraps back around. If the number gets larger than largest allowed number, the integer variable wraps to a negative number. If the value number gets smaller than the smallest allowed number (around negative two billion), the integer variable wraps back into being positive. In any case, the returned number will be an error.
- The name of a strategy created in IRD can incorporate only 194 characters. If you create the `Simple Routing` script object (or rename an existing one) in Configuration Manager with the 254-byte name length, such an object will be displayed in the `IRD Strategies` list, but its `*.rbn` file is not accessible for any kind of operation.
- The names of List objects items and names of this items keys must consist of alphanumeric values and/or underscores (_) only.

Creating Strategies

Note: Starting with 7.6, IRD installation gives the option of configuring a custom security banner message that displays when users log into IRD. For more information, see the chapter on configuring by using the wizard in the *Universal Routing 8.1 Deployment Guide*.

After you obtain all of the routing details and requirements for your contact center, you can translate these details into strategies. You create a strategy by placing in the Interaction Routing Designer (IRD) Routing Design window objects that represent different types of interaction handling.

IRD enables you to design strategies to handle all customer circumstances. For example, if you need a database lookup in the strategy in order to obtain different types of information, you can use the Database Wizard object (see [page 108](#)) in the strategy to obtain the information.

To provide greater flexibility in strategy design and use, you can use variables (see [page 92](#)) when specifying parameters for certain objects, functions, treatments, and expressions. This enables you to create variable-driven strategies. For example, using the Assign object (see [page 120](#)) with a variable enables you to collect data, and then route based on that data.

After you create, save, and compile your strategy, the strategy is saved in two parts, as a Script object and as an *.rbn file. The Script object is stored in the Configuration Database within the Configuration Layer. The *.rbn file is stored in the location that you specify in the Save (or Save As) dialog box (see “Saving Strategies” on [page 60](#)). When strategy is saved, URS automatically stores the strategy into its memory, so it can access it any time.

Note: When you save a new strategy, the dialog box gives the option of saving to a predefined subfolder of the Scripts folder or to a predefined subfolder of any other folder of Script type created in Configuration Manager.

Before loading the strategy onto a routing point for use in production, be sure to test it for problems (see “Testing Strategies” on [page 34](#).)

URS Internal Queues and Performance

URS internal queues are designed to enhance performance and reduce memory requirements. Instead of supporting a queue of interactions for every target (potentially for every Agent or Place), information about targets is shared among interactions. This feature achieves a skills-based routing performance that is on par with group-based routing.

Warning! URS performance degradation can occur if strategies are created in such a way that every interaction has its own set of targets. For example, this can happen if you use a variable in the Selection object in such a way that the content of a variable target is retrieved from the database on the basis of Interaction Data.

Note: For information about configuring routing strategies for voice callback, see the “Designing IVR Scripts and Routing Strategies” chapter in the *Voice Callback 7.1 Deployment Guide*.

Testing Strategies

After you finish creating your strategy in IRD, be sure to thoroughly test it before you use it in production. Examine both the completeness of the design and the elements within the strategy.

Reviewing the Strategy Design

After completing the strategy, review it to determine whether all customer situations are covered. For example:

- If a customer is asked to enter information that is used for routing, have you included a way to handle the situation if the customer enters the wrong information, enters the information incorrectly, or does not have the required information to enter?
- Is the strategy set up to handle all types of services and schedules?
- Can the strategy handle a situation in which a component, such as DB Server, shuts down?

If you are not sure whether all circumstances are covered, create test cases to evaluate the strategy, or ask your administrator to review it. If you have not addressed all possible circumstances in the strategy, some interactions will be routed incorrectly or they will be default routed. Be sure to run the strategy in a lab environment before you use it in production.

Compiling and Checking Integrity

As you design a strategy, you might add, delete, or change objects, targets, rules, statistics, or any other item referenced in the strategy. If more than one person works on a strategy, this possibility becomes even more likely.

IRD provides the Check Integrity tool and its Compile feature to help you validate a strategy. This tool highlights any objects that refer to items that cannot be located.

Check Integrity Mode

When IRD is in Check Integrity mode, if a strategy contains an invalid reference, the **Strategies** list in the IRD main window displays a red X over the strategy’s icon.

Then, if you select a strategy that contains an invalid reference, the Details tab displays the name of the invalid reference and provides information to help you correct the problem (see [Figure 5](#)).

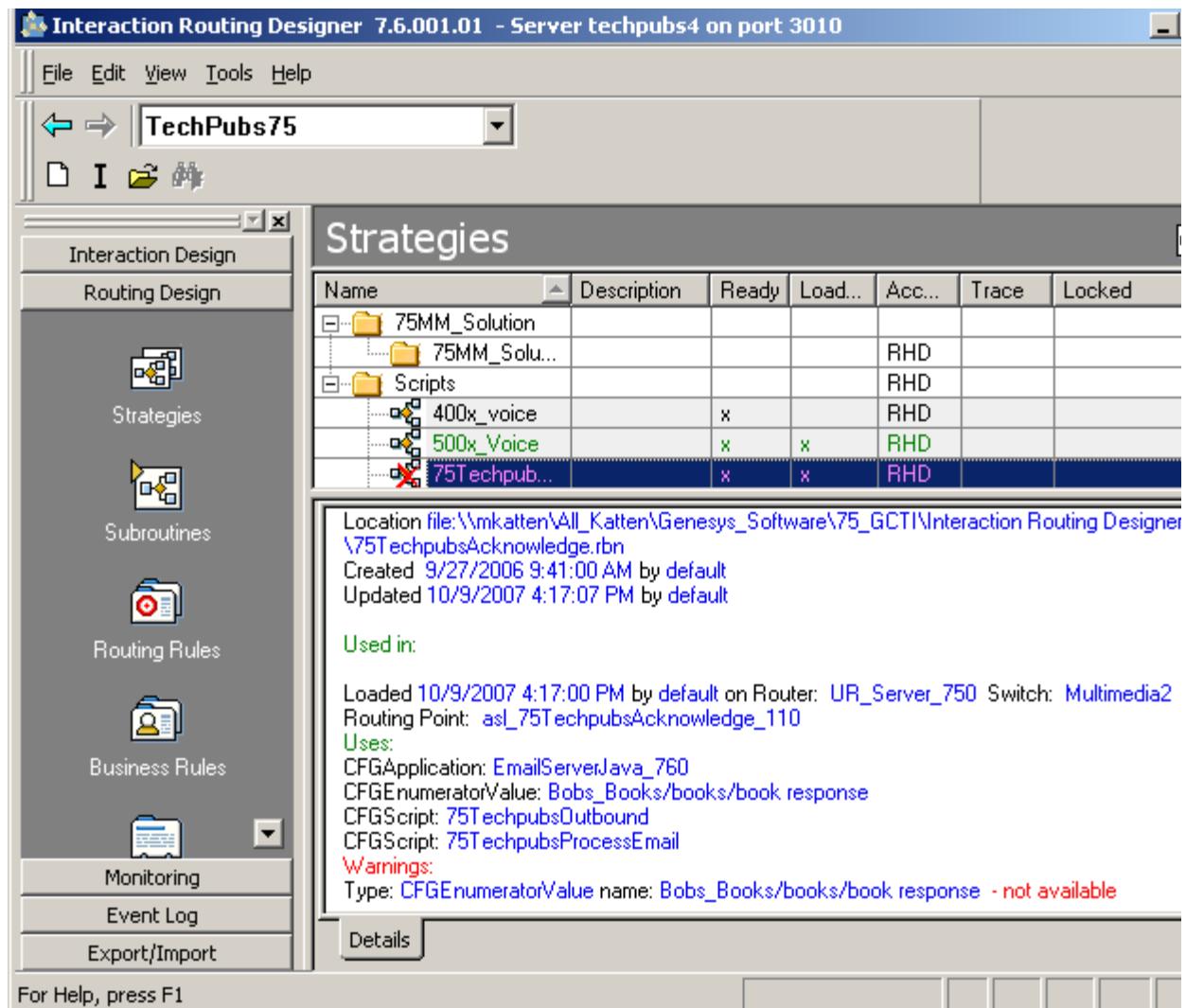


Figure 5: Strategies List Details Tab

Note: Although the Check Integrity mechanism detects when an unknown object type is used in strategy, a strategy with such an error is not marked with a red X and no information is displayed in Details tab. Instead, IRD displays the appropriate error message each time a corrupted strategy is opened or when Check Integrity is performed.

To switch between Normal mode and Check Integrity mode:

- From the View menu, select the mode you want.

To check the integrity of a strategy that is open in the Routing Design window:

- Switch to Check Integrity mode.

2. Open the strategy that you want to check.
3. Click the Check Integrity button ().

The Check Integrity tool highlights any objects in the strategy that contain invalid references. For more information about using the Check Integrity tool, see the *Universal Routing 8.1 Interaction Routing Designer Help*.

Note: Disable Check Integrity mode when you are updating permissions for a large number of objects. If Check Integrity mode is enabled, it could take a long time to update permissions.

Warning! Check Integrity may indicate that it cannot find an object in the database, such as an agent group, that is used in the strategy. You can still save, compile and load such a strategy. However, you should review the list of errors to make sure that the objects referenced by the strategy actually exist.

The Debugging Tool

In your test environment, you can debug a strategy or subroutine by performing the following steps:

1. Load the strategy on a routing point.
2. From the Loading list, right-click the routing point.
3. Select the Debug option from the shortcut menu (see [Figure 6](#)).

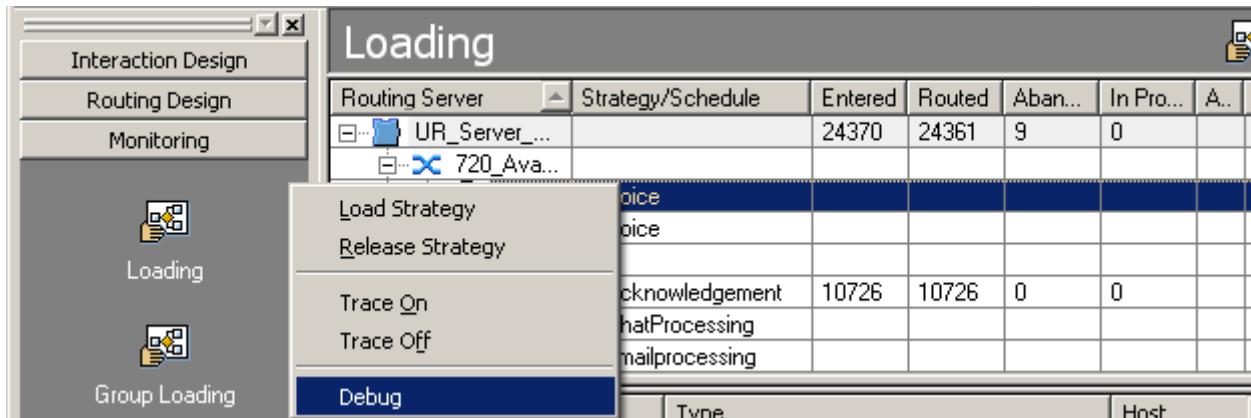


Figure 6: Selecting Debug in the Loading List

Note: For detailed information about using the strategy debugger, see the *Universal Routing 8.1 Interaction Routing Designer Help*.

After the debug session is started, a dialog box prompts you to enter debugging parameters.

Note: The debug functionality is available only for strategies. You cannot use the debug functions for business processes that appear in the Interaction Design window.

After you set debug parameters and click **OK**, the strategy opens with the Entry object highlighted. Use the debug buttons (see [Figure 7](#)) to step through the strategy object by object.



Figure 7: Debug Buttons

As you do so, IRD passes the request to URS, which executes all functions in the current object. While executing an object, URS might request that you enter certain information that it cannot obtain for itself. For example, the debugger does not evaluate agent readiness. If the strategy is designed to send an interaction to the agent with the longest time in the Ready state, you must tell the debugger which agent that is (see [Figure 8](#)).

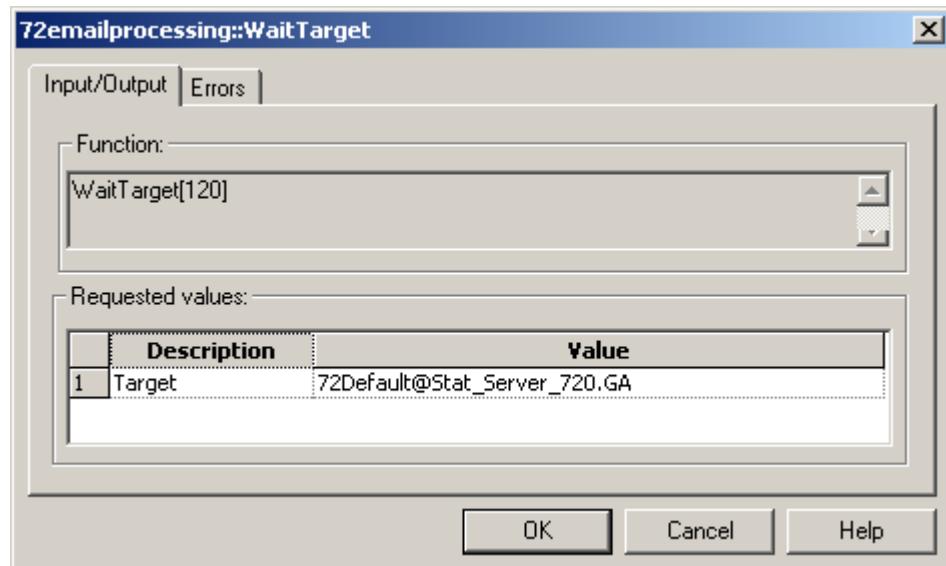


Figure 8: Debugger—Input/Output Tab

You can also click the **Errors** tab and simulate an error to check error handling.

When it has finished executing an object, URS notifies IRD, and suspends action until you issue the next step command. Debugger closes after you reach the end of a strategy, or when you click the Stop debug button.

Note: At any time during a debug session, you can view and edit the content of the variables that are used in the strategy.

Loading a Strategy

You can load a strategy on one or more routing points from the Loading list (see [Figure 9](#)), or you can use Group Loading to load the strategy onto all routing points from some routing point group in one operation. You can also load a strategy by loading a schedule to a particular URS (see “Schedules” on [page 72](#)). In addition, as described in the *Universal Routing 8.1 Interaction Routing Designer Help*, when two or more URS instances run in load sharing mode through Load Distribution Server, IRD synchronizes strategy loading so you do not have to load/unload the same strategy on every URS for each Routing Point in IRD’s Monitoring view.

Note: Non-voice routing strategies that are used in business processes (see [page 153](#)) are loaded on virtual routing points using a special wizard. For more information about this wizard, see the *Universal Routing 7.6* (or later) *Business Process User’s Guide*. The information in this section pertains to loading voice routing strategies only.

After you test and debug your voice routing strategy, and are ready to use it, load it onto a routing point as follows:

1. Click the Monitoring shortcut bar in the IRD main window (see [Figure 6](#) on [page 36](#)).
2. Click Loading.
3. Expand the applicable URS and the switches listed under it to view their routing points.
4. Right-click the desired routing point and select Load Strategy from the shortcut menu. [Figure 9](#) shows strategies loaded in Monitoring view.

Note: In order to edit the strategy, you must first unload it.

Monitoring Strategies

You can use the Loading list to monitor interaction handling on routing points or URSSs. In addition, you can use the Trace View window to view detailed information about the number of interactions that go to each object for all interactions that are routed by a strategy (even if the strategy is loaded on multiple routing points) or for all interactions on a single routing point.

IRD Monitoring View

In IRD Monitoring view, click the Loading button to open the Loading list, from which you can monitor a URS and its routing points as well as load a strategy on a routing point (see [Figure 9](#)).

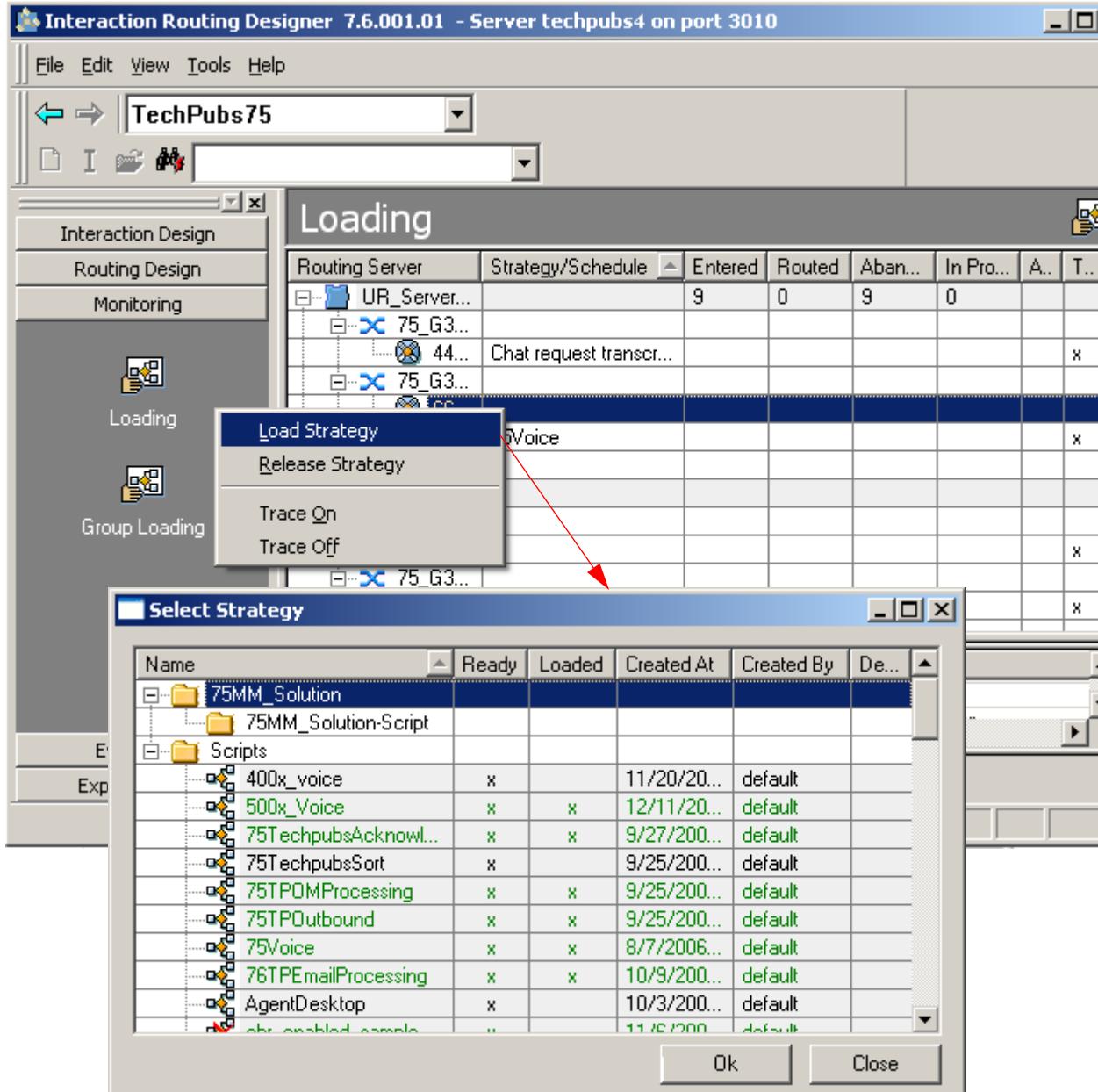


Figure 9: Loading a Strategy in Monitoring View

As shown in [Figure 9](#), the columns in the Loading list show the number of interactions at each of the following stages:

- Entered
- Routed

- Abandoned
- In Process

Trace View

The Trace View window shows interaction counts and percentages for each object in an active strategy or subroutine.

URS collects monitoring data, and then provides it to IRD for display through a shared Message Server. To enable the collection of monitoring data, the strategy, subroutine, or routing point must be marked as “traceable.”

To mark the strategy, subroutine, or routing point and open the Trace View window:

1. Right-click a routing point in the Loading list, a strategy in the Strategies list, or a subroutine in the Subroutines list and select **Trace On** from the shortcut menu.
2. Right-click again and select **Trace View**.

The Trace View window appears (shown in [Figure 10](#)), showing the number of interactions that have reached each object since you selected **Trace On**.

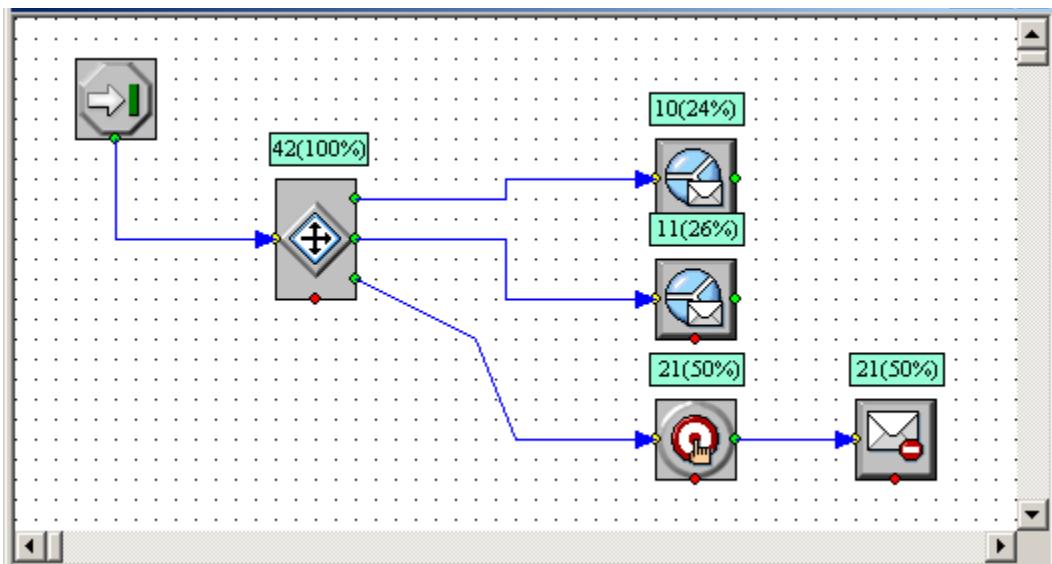


Figure 10: Trace View Window

To clear these counts and make URS stop counting:

1. Right-click the routing point, subroutine, or strategy.
2. Select **Trace Off** from the shortcut menu.

To restart the interaction counts from zero:

1. Right-click the routing point, subroutine, or strategy, and select **Trace Off** from the shortcut menu.
2. Right-click again, and select **Trace On** from the shortcut menu.

3. Right-click again, and select `Trace View` from the shortcut menu.

You can use the `Trace View` window to monitor at the routing point level, as shown in Figure 9 on [page 39](#), or the strategy/subroutine level. The following sections summarize each level.

Monitoring Routing Points

The interaction counts for a routing point that are shown in the `Trace View` window begin from the moment:

- You select the `Trace On` command for the routing point.
- A strategy is loaded on the routing point if `Trace On` was selected for this routing point when you loaded the strategy.

Only interactions that are on the monitored routing point are counted. The numbers and percentages shown in the `Trace View` window exclude:

- Interactions that are handled by the same strategy but located on different routing points.
- Interactions located on this routing point, that are being handled by a strategy other than the loaded one. This can occur if an interaction has been transferred to a different strategy (but has not been moved to a different routing point) by the `Switch-to-Strategy` object.

Monitoring Strategies or Subroutines

The interaction counts for a strategy or subroutine that are shown in the `Trace View` window begin from the moment the `Trace On` command is executed for the strategy or subroutine.

Interactions that are located on all routing points on which this strategy is loaded are counted, even if they arrive from a different strategy, such as when using the `Switch-to-Strategy` object (see [page 347](#)).

Example: Strategy A is loaded on 5 routing points and Strategy B is loaded on 3 routing points. Strategy B contains a jump to Strategy A.

In this case, the `Trace View` window for Strategy A displays interaction counts for all 8 routing points. The `Trace View` window for Strategy B shows interaction counts only for the 3 routing points on which Strategy B is loaded.

The window caption of the `Trace View` window displays the name of the strategy that you are monitoring. It does not show the names of the routing points on which the strategy is loaded or of any strategies that use a `Switch-to-Strategy` object to send interactions into the monitored strategy.

Sharing Strategies

Because strategies must be created carefully and it can take a long time to design them, especially when the needs of the contact center are complex, you might want to share strategies with other people—either in the same routing environment as you, or in a different environment or network.

Sharing strategies enables you to:

- Take someone’s previously created strategy and adapt it to your needs without having to recreate every object.
- Test a strategy in a different environment while the strategy is still in development.

A strategy can be made up of various types of objects—for example, subroutines, routing rules, business rules, attributes, interaction data, and statistics. Therefore, if you want to share a strategy, you cannot provide just the strategy’s *.rbn file to the other user because the *.rbn file does not contain these objects. You must also provide all of the objects that the strategy requires.

The process for sharing a strategy varies depending on whether you are sharing the strategy with someone within the same environment or in another environment.

Warning! A strategy or subroutine (*.rbn file) should not be shared by multiple Configuration Servers. This occurs if you import a strategy or subroutine from the storage location used by one Configuration Server into that used by another Configuration Server without changing the location setting of the *.rbn file in IRD. When two or more Configuration Servers are involved, avoid using the same strategy storage location for strategy storage.

For example, if you have a lab environment and a production environment, always keep the two strategy development environments separate. If you make changes to the strategy in one environment, and want the other environment to include the same changes, export the strategy from the environment in which you made the changes, and then import the strategy into the other environment. For instructions on importing and exporting strategies, see “Exporting Strategies” on [page 46](#), and the *Universal Routing 8.1 Interaction Routing Designer Help*.

Sharing Strategies Within the Same Environment

When sharing strategies with people in the same environment, you have to provide the other users only with access to all of the objects that the strategy uses.

You set up access permissions for all objects through Configuration Manager. For more information, see the *Framework 8.1 Configuration Manager Help*.

If more than one user is going to be working with the same strategy, make sure that all users have access to the storage location for the saved *.rbn file. (This *.rbn file can be on a shared drive on a personal computer, on a network drive of the LAN, or in the Configuration Server database.) When you specify a path in IRD for the storage location, do not enter the local path for a specific computer; instead, enter the complete path name, including the network information.

For more information, see the “Sharing Within a Network” topic in the *Universal Routing 8.1 Interaction Routing Designer Help*.

Sharing Strategies With a Different Environment or Tenant

When sharing strategies with people in a different configuration environment or tenant, you must provide the other user with all of the objects that the strategy uses, so that the other user can rebuild the strategy in the new environment.

These objects include:

- The *.rbn file for the strategy, which contains the graphical and logical information about a strategy.
- The *.rbn files for all subroutines within the strategy.
- The strategy configuration objects (also called Script objects), which store strategy information such as the compiled bytecode, size, path, and other strategy properties.
- The *.rbn file called if the Switch-to-Strategy object is used.
- All Transaction objects that are used in all of the strategies and subroutines you want to package. This includes objects such as routing rules, interaction data, business rules, attributes, statistics, schedules, and lists.
- All Field, Format, and Table Access configuration objects in the Configuration Database, if a strategy includes a Database Wizard object.

Strategies and their associated objects can all be packaged and unpackaged using an automated process (except for the configuration objects that are used for database access).

Sharing Strategies Using the Automated Process

Interaction Routing Designer provides you with commands that automate the process of sharing strategies with a different environment or tenant.

This automated process packages all objects created by IRD. It does not, however, package objects that are created through other applications (usually Configuration Manager)—for example, Person, Place, DN, Field, Format, Table Access, and so on. If you want these items in the new environment, you must create them manually, using the appropriate application.

Locking Strategies and Subroutines

IRD 7.6 (or later) can lock a strategy or subroutine to warn others that it is in use (for example, when it is in the process of being edited). When a user tries to open a locked strategy or subroutine, overwrite it with another one, or restore a previous version, IRD generates a warning message notifying that the strategy is locked. If necessary, you can override a lock set by another user. The name of the host computer from which the lock was set appears on the IRD main window's Details pane, below the Strategies or Subroutines list. For more information on this feature, see the *Universal Routing 8.1 Interaction Routing Designer Help*.

Note: Even though a strategy may be locked by another user, you can still modify any reusable objects (such as subroutines, routing rules, interaction data, and so on) used in a locked strategy without IRD issuing a warning.

Exporting/Importing Objects Between Strategies

You can export the properties of a strategy object to a text file and then import the object into another strategy with the option of editing the text file before inserting the object. The procedure is summarized in the following steps. For step-by-step instructions, see the *Universal Routing 8.1 Interaction Routing Designer Help*.

To export the properties of a strategy object in the Routing Design window:

1. Right-click the object and select `Export object` from the shortcut menu (see [Figure 11](#)).

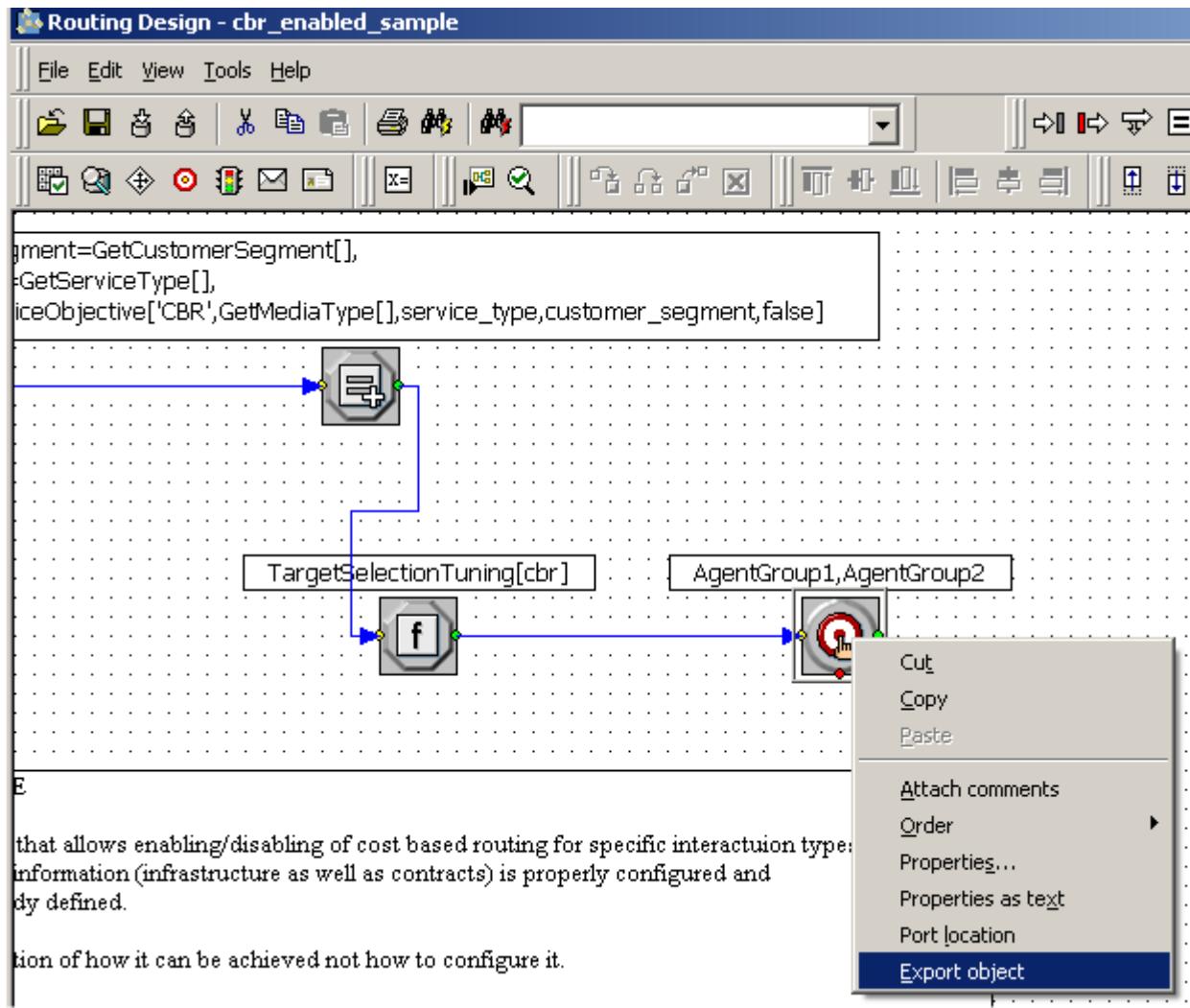


Figure 11: Exporting Strategy Object Properties

2. In the resulting Select file for export dialog box, the *.kvp file type is already selected. Name the file and click Save.
You can edit the file outside of IRD and edit it or you can edit it from within IRD as described below.
3. Open the Routing Design window for the strategy that will import the object.
4. Right-click in the white area of the Routing Design window and select Import object from the shortcut menu. The Import strategy objects dialog box opens.
5. Click Load Data in the Import strategy objects dialog box. The Select import file dialog box opens.
6. Select the *.kvp file to import and click Open. The Import strategy objects dialog box shows the strategy object's properties. [Figure 12](#) shows an example imported text file after clicking the Show gui checkbox.

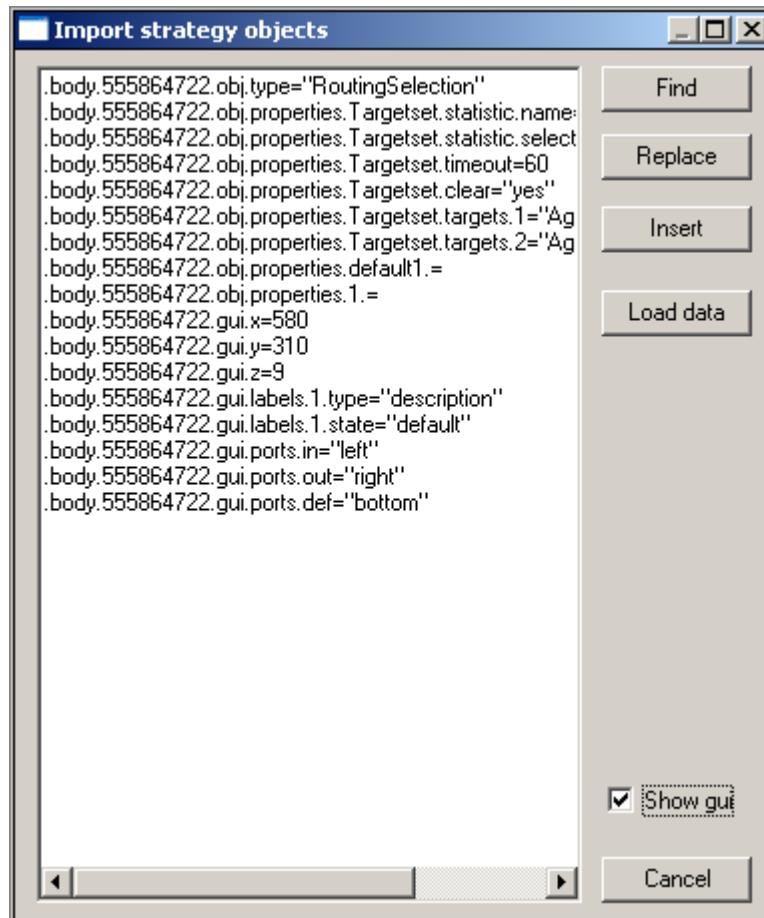


Figure 12: Importing Strategy Objects

7. Inside the dialog box, you have the following options:
 - Modify the object's properties. Genesys strongly recommends that you limit modifications to the name of configuration objects and/or expression strings that they contain.
 - Click Replace to open a dialog box in which you can find and replace strings.
 - Click Find to open a dialog box in which you can search for strings.
8. When ready, click Insert to place the object in your strategy. The object appears in the Routing Design window workspace.

Exporting Strategies

Note: Also see “Exporting/Importing Objects Between Strategies” on [page 44](#).

You can export strategies in two ways:

1. Using the `Export To File` menu command available from the IRD main window `File` menu or via a strategy shortcut menu (as described in this section).
2. Using a dedicated `Solution export` view (see “Migrating Strategies and Other Objects” on [page 54](#)).

Notes: To export just a single strategy/subroutine, it is more convenient to use the `Export to File` menu command.

Use `Solution export` view to export a group of strategies or subroutines, or to export only objects (such as routing rules, business rules, interaction data, statistics, or macros) without strategies that use them.

IRD’s `Export To File` command enables you to export (*package*) strategies using one of three formats: `archive`, `open`, or `text`. `Solution export` view allows one more format: `native` (see Figure 17 on [page 56](#)).

Export to File Menu Command

To export using the `Export to File` menu command:

1. Select a strategy.
2. Select `Export to File` from the `File` menu or from the shortcut menu. A dialog box opens (see Figure 13).

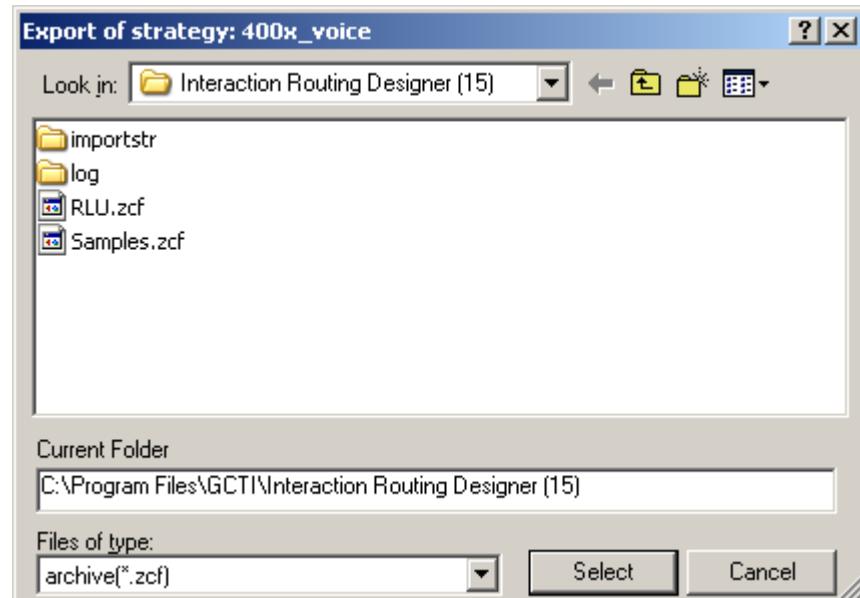


Figure 13: Export of Strategy Dialog Box

3. Keep the entry in `Current Folder` or select a new folder.

4. Under **Files of type**, select archive, open, or text.

5. Click **Select**.

IRD creates the export file in the specified folder. The file will have the same name as the strategy being exported. If a file with such name already exists in selected folder, it will be overwritten. [Figure 14](#) shows example export files for all three formats.

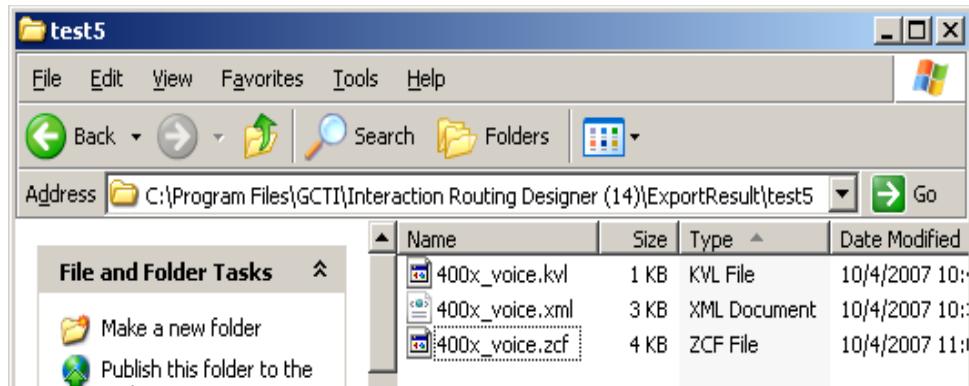


Figure 14: Example Export Files

You can export one strategy at a time. Alternatively, when working in IRD's **Interaction Design** window, you can export all strategies belonging to a business process in a single operation as described in the topic on exporting and importing business processes in the *Universal Routing 8.1 Interaction Routing Designer Help*.

Exporting Using Archive (*.zcf) Format

When you export a strategy in archive (*.zcf) format, IRD creates two files:

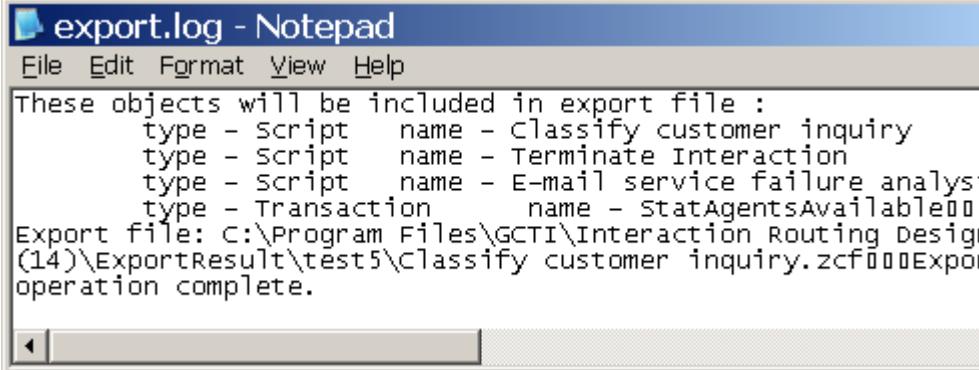
1. The export file itself, which is a *.zcf file containing all of the strategy information. Only objects created in IRD are packaged; this includes the strategy itself, subroutines, business rules, routing rules, attributes, interaction data, and statistics. The name of the .zcf file matches the name of the strategy with the addition of the .zcf extension.

Note: Access permission can affect the ability of another person to unpackage the strategy, if the person does not have the appropriate permission level for a strategy object. Therefore, when providing a strategy to another user, warn the user of the permission levels that have been set for the objects.

The folder structure for strategy objects is not stored. For example, if the strategy's routing rules were stored in a special folder within the **Transactions** folder in Configuration Manager, this special folder would not be created when the file was unpackaged. Instead, all routing objects would be placed at the top level of the **Transactions** folder.

Warning! Do not change the name of the *.zcf file. Changing the name of this file will prevent the strategy from being imported.

2. A log of the exporting operation, which is an export.log file. You can use the log file to gauge the success of the export operation, such as which objects were included in the *.zcf file and which objects failed to export (see [Figure 15](#)).



```
export.log - Notepad
File Edit Format View Help
These objects will be included in export file :
    type - Script      name - Classify customer inquiry
    type - Script      name - Terminate Interaction
    type - Script      name - E-mail service failure analysis
    type - Transaction   name - StatAgentsAvailable
Export file: C:\Program Files\GCTI\Interaction Routing Designer (14)\ExportResult\test5\Classify customer inquiry.zcf
Export operation complete.
```

Figure 15: Example Export.log File

Note: IRD does not create an export.log file when you select open (*.xml) or text (*.kvl) format.

IRD places the export.log file in the IRD working directory (see [page 59](#)). Any existing export.log file is overwritten. Objects in a strategy (*.rnb file) that do not have counterparts stored in the Configuration Database—objects internal to the strategy—are not included in the export.log file. However, they are included in the package because they are part of the *.rnb file.

There are several reasons why IRD might not be able to package an object. For example, the object might have been deleted or renamed in Configuration Manager or you might not have access permission for the object.

The following is an example of an export.log file, for a strategy called MovingStrategy:

```
These objects will be included in the export file MovingStrategy.lst
    type - Script name - MovingStrategy
    type - Transaction - LoadBalance1
    type - Transaction - LoadBalance2
Exporting operation complete.
```

Note: Every time you package a strategy, the `export.log` file is overwritten. If you want to save this file for future reference, move the file to a location other than the working directory of your IRD application.

For step-by-step instructions on using an automated process to package a strategy, see the *Universal Routing 8.1 Interaction Routing Designer Help*.

Note: The `bytecode` option determines whether URS includes bytecode configuration data when you create a `.zcf` file for export. For more information, see “`bytecode`” on [page 695](#).

Importing (Unpackaging) a Strategy

You can import strategies in two ways:

1. Using the `Import From File` menu command, which is available from the IRD main window `File` menu (as described in this section).
2. Using a dedicated `Solution import` view (see Figure 20 on [page 58](#)).

For input data, you will normally use the result of some previous export operation.

Notes: To import the results of a strategy export created with the `Export To File` menu command (see [page 47](#)), use the `Import From File` menu command from the IRD main window `File` menu as described in this section.

To import the results of an export created with the `Solution export` view, use `Solution import` view (see Figure 20 on [page 58](#)).

Making Strategies Ready for Immediate Execution

Whether or not the imported strategies contain bytecode (and will therefore be ready for immediate execution) depends on the presence of bytecode in the `*.zcf` file. You may want to set the IRD `bytecode` option (see [page 695](#)) to `true` in order to guarantee that a `*.zcf` file will contain the bytecode for exported strategies.

Import Operation

When you import a `*.zcf` file, all objects are created and stored in the appropriate location, according to their object type. This process creates a subdirectory called `importstr` in the IRD working directory, in which the strategy is stored until the unpackaging operation is complete.

Note: When you export a strategy into an `*.xml` or `*.kvl` file using the `Export to File` menu command, only strategy and/or subroutine objects are exported. All other objects are not exported, therefore importing those files cannot create original objects.

Access Permissions

Your ability to import a strategy depends on the access permissions that have been set on the objects being unpackaged, and on your the access level. If the objects in the package have permission levels set to `Administrator`, you do not, you cannot completely unpack the strategy. This prevents unauthorized users from accidentally overwriting existing objects with ones that have the same names as those in the package. For more information, see “Strategy Import Rules” on [page 52](#).

Log File

An `import.log` file is generated for when the import operation for an `*.zcf` file is complete (importing `*.xml` or `*.kvl` files does not generate a log file).

The log file lists those objects that were successfully unpackaged and those that could not be unpackaged. (Objects in a strategy `*.rbn` file) that do not have counterparts stored in the Configuration Database—objects internal to the strategy—are not included in the `import.log` file. However, they are included in the package because they are part of the `*.rbn` file.)

There are several reason why objects might not be unpackaged. For example, you might not have the appropriate permission level for an object, or the connection to Configuration Server might be broken.

The following is an example of an `import.log` file, for a strategy called `MovingStrategy` that has been imported to an environment that already contained objects with the same names:

```
Importing operation for script - MovingStrategy
These objects were rewritten:
Script - MovingStrategy
Rule - LoadBalance1
Rule - LoadBalance2
These objects were imported into the database:
Script - MovingStrategy
Rule - LoadBalance1
Rule - LoadBalance2
Importing operation complete.
```

For instructions on using the automated process to unpack a strategy, see the *Universal Routing 8.1 Interaction Routing Designer Help*.

Notes: Every time you import a strategy, the `import.log` file is overwritten. If you want to save this file for future reference, move it to a location other than the working directory of your IRD application.

When importing a strategy into a new environment, make sure that the strategy is set up with the correct locations for the environment and Tenant, including its storage location. This is especially important if you are moving a strategy from a lab environment to a production environment. If the strategy is not set up properly, and if the `*.rbn` file is shared between these environments, any changes that are made in one environment will affect the other.

Strategy Import Rules

If none of the rules described in this section are violated, IRD imports the strategies and provides a successful completion message.

Note: If you do not set the IRD `bytecode` option as described on [page 695](#), after importing a strategy (`.zcf` or `*.rbn` file), you must open, save, and recompile it in order for it to function properly. If an imported strategy contains a subroutine, you must also open, save, and recompile the subroutine.

If you are unable to import a strategy, do the following:

1. Review the following list of rules, to determine what the problem might be.
2. View the `import.log` file, to determine which objects were successfully unpackaged and which objects were not.

Note: Only objects that are used in the strategy and that are stored in the Configuration Database appear in the `import.log` file.

Cancellation Rules

When IRD cancels the process of importing a strategy it displays a cancellation message and makes no changes to the new environment.

- For strategy and subroutine objects, the cancellation occurs in the following situations:
 - You do not have permission (access rights) to create these objects.
 - Some of the objects already exist in the new environment and you do not have permission to change at least one of them.
 - At least one `*.rbn` file with the same name already exists in the current strategy storage location, and you do not have the strategy or subroutine that references this `*.rbn` file. IRD considers that such

- *.rbn file is probably referenced by some strategy from other environment. Importing is canceled to prevent potential damaging of others environments.
- At least one of these objects already exists in the current environment and the strategy or subroutine is open for editing in IRD.
- For business rules, routing rules, attributes, interaction data, and statistic objects, the cancellation occurs in the following situations:
 - The strategy being imported contains at least one of these objects and you do not have permission (access rights) to create them.
 - Some of the objects already exist in the new environment and you do not have permission to change at least one of them.

Warning Rules

If some of the objects being unpackaged already exist, and you have permission to change all of them, a warning dialog box appears. You have two options: cancel or confirm the operation.

- If you cancel the operation no changes are made to the environment.
- If you confirm the operation, all existing objects that share the same name as the objects being imported are overwritten.

Note: You cannot selectively overwrite some existing files. If you do not want to overwrite all of the existing files listed in the dialog box, cancel the operation. Selective overwriting is available through [Solution import view](#) (see [page 58](#)).

As mentioned in “Cancellation Rules” on [page 52](#), if you do not have permission, the operation is cancelled.

- For strategy and subroutine objects, a warning dialog box appears, regardless of whether any of those objects includes its corresponding *.rbn file in the strategy storage location. The warning dialog box includes a list of objects that will be overwritten if you confirm the operation. If you confirm the operation, a successful completion message appears when the operation is finished.

If the existing *.rbn file that has the same name as one of the *.rbn files being imported is not stored in the strategy storage location, IRD will not overwrite it.

- For business rules, routing rules, attributes, interaction data, and statistic objects, a warning dialog box appears, and includes a list of objects that will be overwritten if you confirm the operation. If you confirm the operation, a successful completion message appears when the operation is finished.

Warning! IRD does not distinguish existing objects that are used in strategies loaded on a routing point from those objects that are not in a loaded strategy. If the user who is importing a strategy has the necessary permissions, the objects will be overwritten. This can affect interactions that are currently being processed.

Migrating Strategies and Other Objects

The IRD main window has Export Solution and Import Solution views.

Solution Export View

Notes: Use Solution Export view as described in this section to export a group of strategies or subroutines. You can also use this view to export only objects (routing rules, business rules, interaction data, statistics, macros, and so on) without the strategies that use them.

To export a single entire strategy, the Export to File menu command (see [page 47](#)) is more convenient.

The Solution export view does not automatically display updates made from outside your instance of IRD (for example, changes to configuration objects, or the switching on or off of the Check Integrity function). To update the view, right-click anywhere on the List pane, and then select Refresh.

You can leave the Solution export view and use other IRD functionality without reactivating the Solution export view. However, if you return to the Solution export view, the window is still inactive, and you must clear it. (Refreshing the Solution export view, or closing IRD and restarting it, also clears the Solution export view.)

To access Solution Export view:

1. Click the Export/Import shortcut bar in the IRD main window.
2. Click the Solution export icon.

[Figure 16](#) shows an example Solution Export view with a Scripts folder (for strategies and subroutines) and Transactions folder (for objects used by strategies).

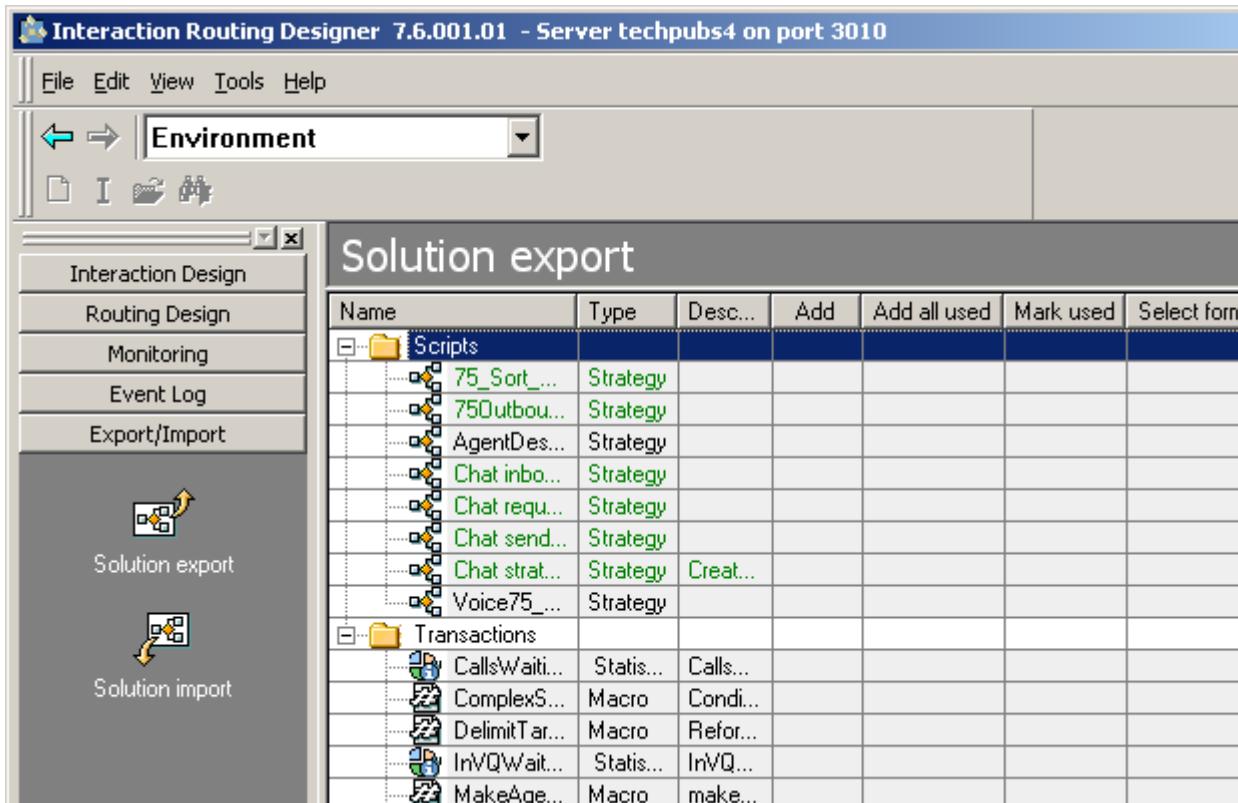


Figure 16: Example Solution Export View

The Solution Export view appears similar to views used for strategies, routing rules, business rules, and so on. It has the same tree-like structure. The main difference from other views is that the Solution export view lists all object types supported by IRD.

Note: The procedure for performing the export operation for both strategies and strategy objects is summarized in the following steps. For step-by-step instructions, see the *Universal Routing 8.1 Interaction Routing Designer Help*.

1. Mark objects to be exported as follows:

- To include the selected object as well as *all objects used by that object*, use *Add all used object to export list* from the shortcut menu. An x will appear in the *Add all used* column for the selected object. The letter Y will appear in the *Mark used* column for any object(s) that it uses.
 - To add a *single* object of any type, use *Add the object to the export list* from the shortcut menu. An x will appear in the *Add* column.
- [Figure 17](#) shows a strategy selected, but you can add any of the object types that appear in Solution Export view.

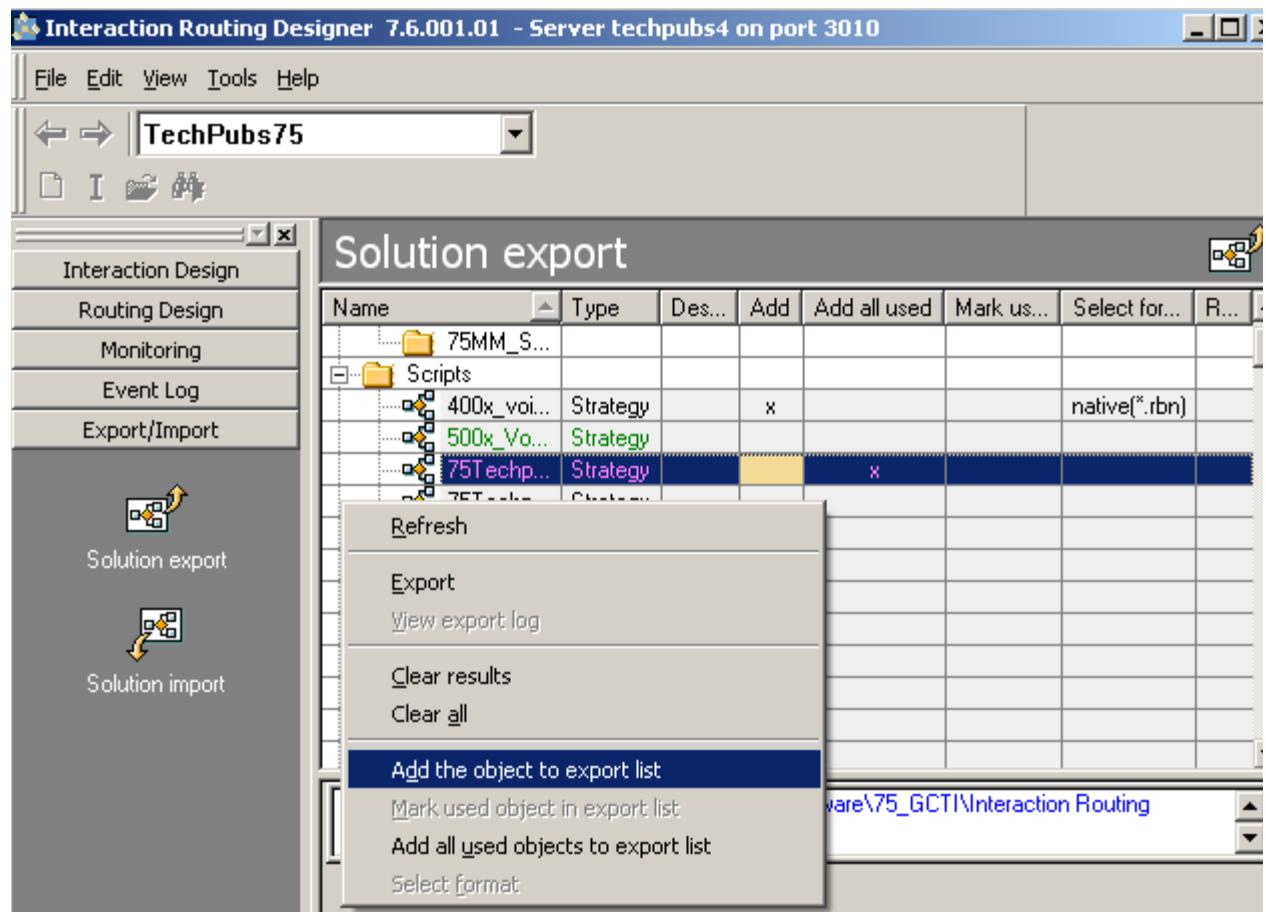


Figure 17: Solution Export View and Shortcut Menu

Note: To put some object and all objects that it uses (analog of *.zcf file content) into an export list, you need to do both of the above-mentioned actions for this object.

2. Once the export list is formed, you can change the format in which objects will be exported. **Figure 18** shows the menu for a strategy or subroutine.

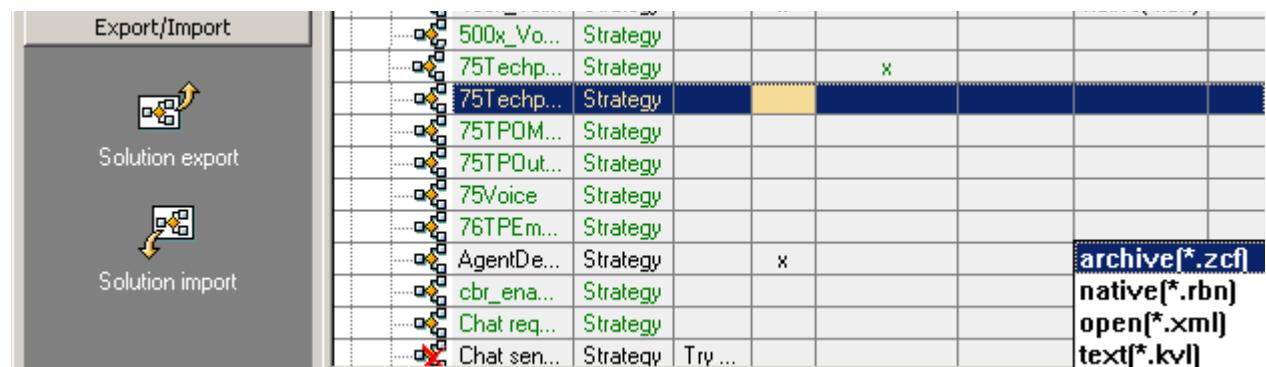


Figure 18: Export Format Menu

Note: Strategies and subroutines can be exported using any one of four formats: archive (*.zcf), native (*.rbn), open (*.xml), and text (*.kvt) format. All other objects are exported in text (*.kvlt) format.

3. Perform the export operation itself by the shortcut menu command Export. You must specify a folder to store the export results.
 - By default, IRD proposes to store export results in a folder with a name composed of the current date/time and located in subfolder <IRD working directory>/ExportResults (see [page 59](#)). You can either accept this default folder or specify another folder.
 - An export completion dialog provides a short information about how successful the operation was and also provides option to see detailed log of operation.
 - The column Result (partially shown as R... in Figure 18 on [page 56](#)) indicates whether every object on the export list was successfully imported.
4. Post operation. Use the context menu command View export log. [Figure 19](#) shows an example log report.

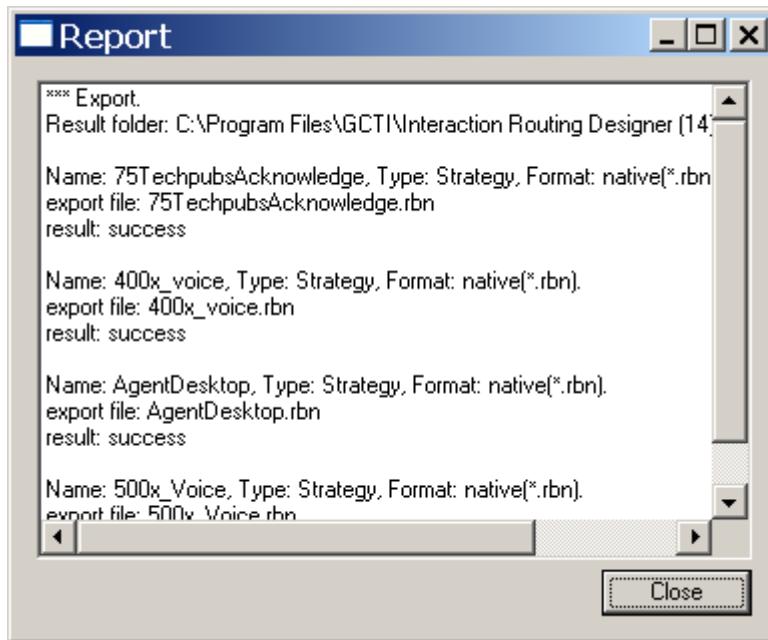


Figure 19: Example Export Log Report

The export log file is placed into the same directory as the export results. To start a new export operation, you must clear the results of the last operation, which is done with two context menu commands.

Solution Import View

This view is similar to `Solution export` view but designed to move IRD objects from a file system to the Configuration Database. The main difference is that `Solution import` view starts out empty. The importing operation is performed with just a few steps:

1. To be imported, the content of a folder created by an exporting operation must be loaded into `Solution import` view. Use the shortcut menu command `Load data` for this purpose.
2. Just like for `Solution export`, you must add objects to list. Use `Add the object to import list` from the shortcut menu for this purpose. If an object is a new configuration, you can adjust its location in the Configuration Database.
3. Perform the import itself through the shortcut menu command `Import` (see Figure 20).

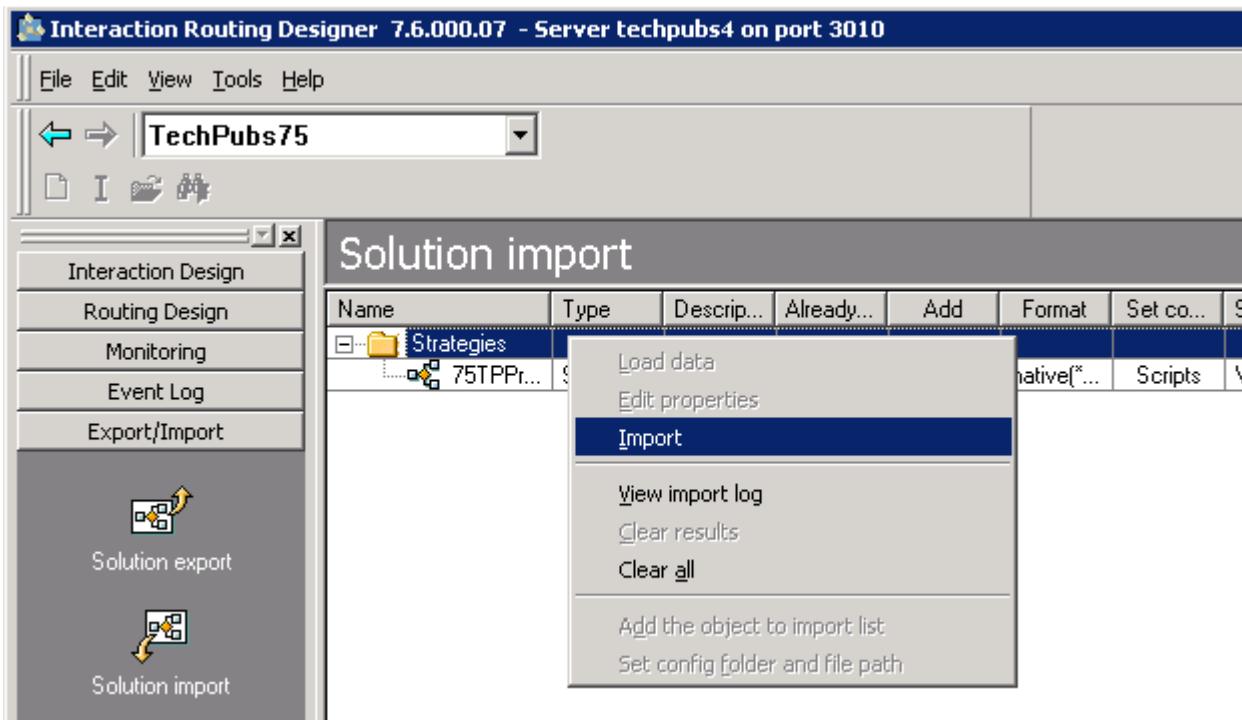


Figure 20: Solution Import View and Shortcut Menu

Note: In order to use `Edit properties` from the shortcut menu, the strategy must be saved in text (*.kvl) format as shown in Figure 18 on [page 56](#).

4. Upon completion, IRD displays a short message about the operation and gives the option to look at a detailed import log.

5. Use the context menu command `View import log`. To start a new importing operation you need to clear the results of the last operation, which is done with one of two shortcut menu commands:
 - `Clear results`, which clears the Result column in the IRD window. The export list remains as it was so it is possible to repeat the same import operation.
 - `Clear all`, which clears both the Result column and the export list.

For more detailed step-by-step instructions on using the `Solution Export` and `Solution Import` views, consult the *Universal Routing 8.1 Interaction Routing Designer Help*.

Modifying Strategies

Note: Starting with 7.6, Routing strategies have warning mechanism. When there is a conflict in opening the same strategy, a message displays warning the user that the strategy is already in use. This mechanism prevents the potential over writing of strategies by others and implements user security. For more information on this feature, see the *Universal Routing 8.1 Interaction Routing Designer Help*.

After creating strategies it is often necessary to modify them. IRD enables you to do this easily. However, keep the following things in mind:

- Deleting or changing the name of any item (strategies, objects, rules, statistics, and so on) that is referenced in other strategies, rules, objects, or functions invalidates the strategy unless you update the object that is using the referenced item.
- Before changing or deleting any object, such as a routing rule, go to the List view for that object to see which strategies include it, and then modify the strategies as needed so that they are not rendered invalid.
- Use the Check Integrity tool to determine whether you have invalid objects in your modified strategy. For information on the Check Integrity tool, see “Compiling and Checking Integrity” on [page 34](#).

Specifying a Working Directory

The Storage Settings tab of the Routing Design Options dialog box enables you to specify a Working directory. This directory contains `log`, `ImportResult`, `ExportResult`, `log`, and `importstr` folders. (To open the Routing Design Options dialog box, select `Routing Design Options` from the `Tools` menu.)

The Storage Settings tab might also contain the following files: `import.log` (after importing), `export.log` (after exporting), `copyas.log` (after a Save As operation), and `save.log` (after saving). For more information about the

Working directory and the Routing Design Options dialog box, see the *Universal Routing 8.1 Interaction Routing Designer Help*.

Loading an Edited Strategy

If you are updating a strategy that is used in production, and if you load the new version to a routing point at which the old version was loaded, URS retains both versions of the strategy in memory and active simultaneously. This enables URS to continue processing any current interactions using the old version, instead of routing them to the default destination. Once all interactions that arrived at the routing point when the old version was loaded are distributed, the older version is erased from URS's memory. All interactions submitted after the new strategy is loaded are processed by this new strategy. For instructions how to load a new strategy on a routing point that already has a strategy loaded on it, see the *Universal Routing 8.1 Interaction Routing Designer Help*.

Note: If you are updating a strategy that is used in production, be sure that you thoroughly test the strategy before loading it to ensure that no errors have been introduced.

Saving Strategies

The `Compile` and `Find` commands do not implicitly save a strategy. To save a strategy, use either the `Save` (for a new strategy) or `Save (As)` command.

Note: IRD checks and gives a warning when saving a strategy or subroutine if it is locked or used by another strategy or subroutine when the integrity checking mode is enabled. Also, IRD does not allow a loaded strategy to be saved when the integrity checking mode is enabled.

Strategy Elements

When you save a strategy in IRD, the strategy source is comprised of the following elements:

- A Configuration Manager object of type `Transaction`. The `Transactions` folder is located under the `Tenant` folder. It can contain one or more of the following items that have been configured in IRD: routing rules, business rules, interaction data, attributes, list objects, macros and statistics.
- A Configuration Manager object of type `Script`. The `Scripts` folder is also located under the `Tenant` folder. It contains the compiled routing strategy code and the path to the `*.rnb` file, which describes the graphical portion of the strategy.

- The graphical portion of the strategy, which is required in order to view and edit the strategy. This is contained in a file with an *.rbn extension. The path to the *.rbn file is referenced in two places: the Configuration Manager Script object, and the IRD Save and Import dialog boxes.

Saving the RBN File

Note: The IRD main window displays the *.rbn file location in the Details tab (see Figure 5 on [page 35](#)).

Do not select the importstr sub-directory of the IRD working directory as a new *.rbn file path location. This directory was designed to serve as a place for creating and keeping the temporary *.rbn files. If you try to save to this directory, an error message appears warning that a strategy with such a name already exists.

When you use the Save (for a new strategy), Save As, Import from File, or Create Copy command, IRD opens a dialog box, in which you have the option of saving the *.rbn file on a network drive or centrally in the Configuration Database. Because the dialog box appears when you perform any of these actions, you can overwrite previous storage settings at any time.

Storing in a Database

If security is a consideration, you might want to use the database storage method to save *.rbn files. For example, if you are a Service Provider, you might not want your subscribers to have access to your corporate servers. In this case, saving the *.rbn file in the Configuration Database is the preferred method.

For information about running a script that creates the database tables in the Configuration Database in which *.rbn files are stored, see the *Universal Routing 8.1 Deployment Guide*.

Default Storage Location by Tenant

Different tenants can have different default storage locations for strategy *.rbn files (graphical portion of strategy), which are then used in the Save dialog box. To set the default storage location up in Configuration Manager:

1. Bring up the properties dialog box for the selected tenant.
2. In the Annex tab, define a strategy section.
3. In the strategy section, define location and path options.

If the default strategy storage location is a network drive, use file for the location option (quotes are automatically inserted). Enter the directory path as the value for the path option.

If the default strategy storage location is the Configuration Database, use `database` as the value for the `location` option (quotes are automatically inserted). A value for the `path` option is not required.

Note: The `Storage Settings` tab of the `Options` dialog box lets you specify a working directory (see [page 59](#)), which is different than the default storage location by tenant.

Using Save As to Rename Strategies

The `Rename` command has been removed from IRD. To rename a subroutine:

1. Select a subroutine (for example, `Sub1`).
2. Create a copy (naming it `Sub2`, for example).
3. Make the necessary changes in `Sub2`.
4. When you are ready to put `Sub2` into a production environment, save it as `Sub1`.

The original subroutine (`Sub1`) is immediately replaced with new subroutine, without causing default routing. In addition, at this point, you still have `Sub2`, because it was not deleted when you made the copy.

5. (Optional) Delete `Sub2`.

Deleting this modified copy of the original subroutine has no effect on routing functionality.



Chapter

2

Interaction Routing Designer Objects

To create strategies in Interaction Routing Designer (IRD), you do not need to be familiar with the underlying language. Instead, you create strategies by working with objects.

Note: This chapter presents reference information only. See the *Universal Routing 8.1 Interaction Routing Designer Help* for step-by-step instructions on building strategies using objects described here.

The information in this chapter is divided among the following topics:

- [Reusable Objects, page 64](#)
- [Strategy-Building Objects, page 74](#)
- [Object Properties Dialog Boxes, page 85](#)
- [Logical Expressions, page 87](#)
- [Variables in Objects and Expressions, page 92](#)
- [The Interaction Design Window, page 98](#)
- [Business Attributes, page 102](#)
- [Using Objects in Strategies, page 104](#)
- [Data and Services, page 107](#)
- [Miscellaneous Objects, page 120](#)
- [Multimedia Objects, page 149](#)
- [Outbound Objects, page 294](#)
- [Routing Objects, page 311](#)
- [Segmentation Objects, page 374](#)
- [SMS Objects, page 386](#)
- [Workflow and Resource Management Objects, page 392](#)
- [Voice Treatment Objects, page 404](#)
- [Access Control, page 418](#)

Reusable Objects

In Interaction Routing Designer (IRD), you can define the following reusable objects and data, which any strategy can then use:

- Strategies
- Subroutines
- Routing Rules
- Business Rules
- Attributes
- Interaction Data
- Statistics
- Schedules
- List Objects
- Macros

Figure 21 shows the buttons for reusable objects (not all buttons shown).

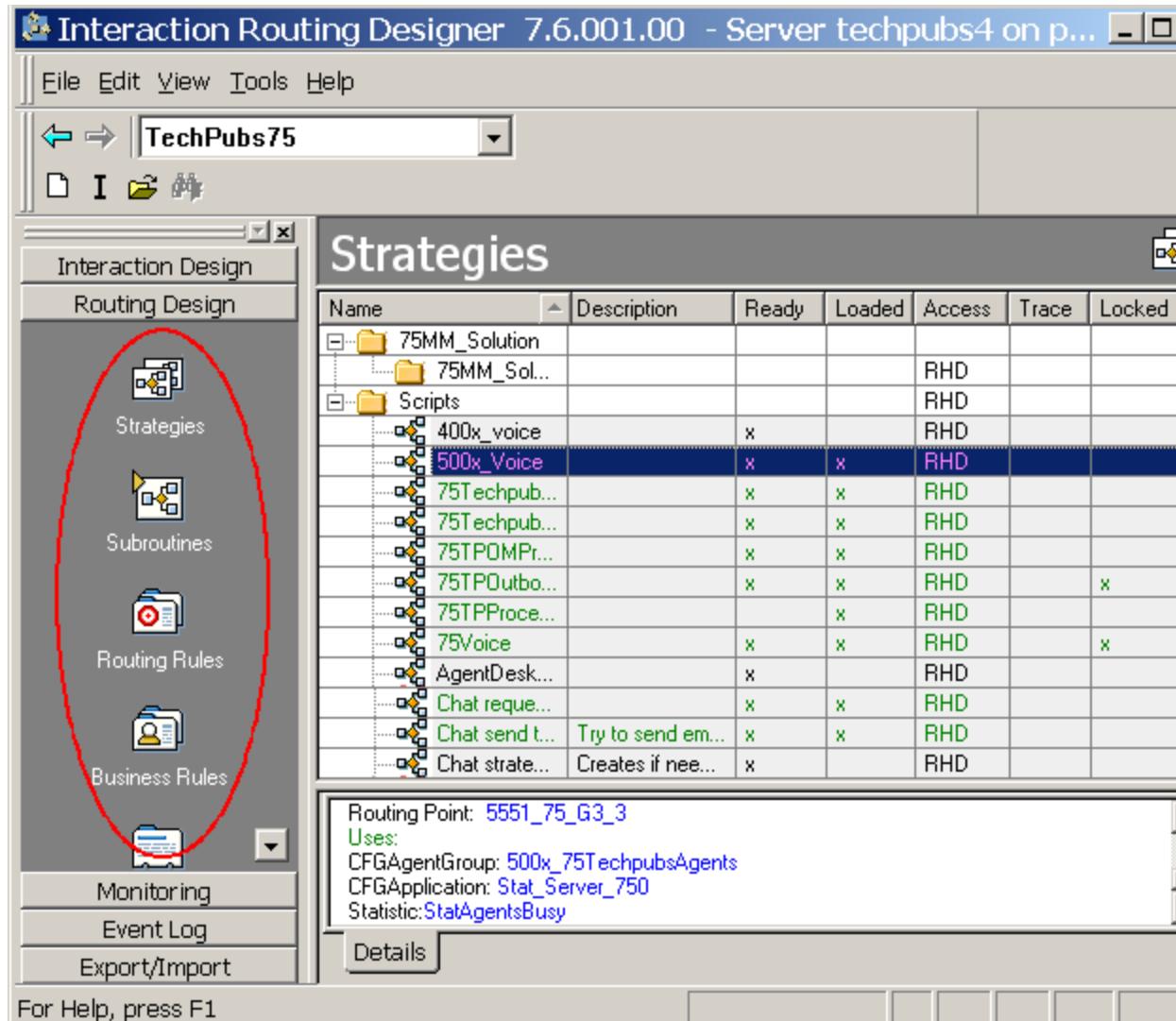


Figure 21: IRD Main Window, Buttons for Reusable Objects

The names of Configuration objects are limited to alphanumeric characters and cannot contain spaces. The objects include Tenant, Field, Format, Script

(strategies, subroutines and schedules), Table Access, and Transactions (routing rules, interaction data, attributes, business rules, list objects, macros and custom statistics).

Summary information about each reusable object is presented in the following sections. For more information, particularly about using these objects in strategies, see the *Universal Routing 8.1 Interaction Routing Designer Help*.

Strategies

Note: The Universal Routing 8.1 installation package places a `Samples.zcf` file in the IRD installation directory. When imported into IRD, this file provides sample strategies. For more information, see the chapter on samples in the *Universal Routing 8.1 Deployment Guide*.

A *strategy* is a set of decisions and instructions that tell Universal Routing Server (URS) how to handle and where to direct interactions under different circumstances (see Figure 3 on [page 29](#)). Within a strategy, you may use subroutines, routing rules, business rules, attributes, interaction data, statistics, lists, and macros.

- You load a routing strategy in IRD Monitoring view by clicking Loading, selecting the switch and routing point, right-clicking, and selecting Load Strategy from the menu (see Figure 6 on [page 36](#)).
- You load the strategies that make up a business process, which is a series of strategies that together make up an interaction workflow, using the Strategy Activation Wizard, as described in the *Universal Routing Business Process User's Guide*. See “The Interaction Design Window” on [page 98](#) for information about creating and using business processes.

Subroutines

Note: The Universal Routing 8.1 installation package places an `RLU.zcf` file in the IRD installation directory. When imported into IRD, this file contains utility subroutines, which can be used for load balancing and other purposes. For more information, see the chapter on samples in the *Universal Routing 8.1 Deployment Guide*.

Subroutines are strategies called from within a strategy or another subroutine. Like other strategies, subroutines can contain any IRD objects or functions. Like other reusable objects, subroutines can be accessed by any strategy. See “Call Subroutine” on [page 121](#) for more information.

In IRD, subroutine names appear in the Subroutines List rather than in the Strategies List.

When a subroutine is called from a strategy, control of the interaction is passed to the subroutine from the calling strategy. The subroutine then instructs URS what to do with the interaction.

If the interaction is not routed to the a destination during the subroutine, control of the interaction returns to the calling strategy at the end of the subroutine. An interaction that is default-routed within a subroutine does not return to the strategy from which it came.

Every execution path within a subroutine must contain an Exit object. If it does not, interactions are routed to the default destination.

When an interaction is routed from a subroutine, any object that is capable of generating a **0008** error code (routing done), if placed after the subroutine (postrouting), generates the error code, **0008**.

A subroutine can be:

- Called at any point during a strategy.
- Used in more than one strategy, making it independent of the strategies that call it.
- Used within other subroutines, offering the capability of creating multiple levels of subroutines.

Note: Unlike a strategy, a subroutine cannot be loaded on a routing point.

Cautions When Using Subroutines

- Be careful when placing subroutines within subroutines. If multiple levels of subroutines are not properly configured, control of an interaction may be passed from one subroutine to another creating an infinite loop without ever routing the interaction to a destination. Also be careful that you do not call the original strategy as a subroutine for itself.
- A strategy or subroutine (.rbn file) should not be shared by multiple Configuration Servers. This occurs if you import a strategy or subroutine from the same storage location used by one Configuration Server into another Configuration Server without changing the location setting of the .rbn file in IRD.

For example, if you have a lab environment and a production environment, always keep the two strategy-development environments separate. If you do make changes to the strategy in one environment and want the other environment to include the same changes, export the strategy from the environment in which you made the changes and import the strategy into the other environment. See the *Universal Routing 8.1 Interaction Routing Designer Help* for instructions on exporting and importing strategies.

- Be careful when changing existing subroutines. Subroutine names and content can be changed even if the subroutine is called by a loaded strategy. If you change the subroutine name without updating the Call Subroutine object that uses the subroutine, a message appears in the output window for the strategy warning that the subroutine cannot be found.

Note: IRD checks and gives a warning when saving a strategy or subroutine if it is locked or used by another strategy or subroutine when the integrity checking mode is enabled. Also, IRD does not allow a loaded strategy to be save when the integrity checking mode is enabled.

Subroutine Input and Output Parameters

When creating subroutines, an **Input Parameters** dialog box allows you to specify the nature of all input parameters. [Figure 22](#) shows the **Subject** dropdown menu in the dialog box.

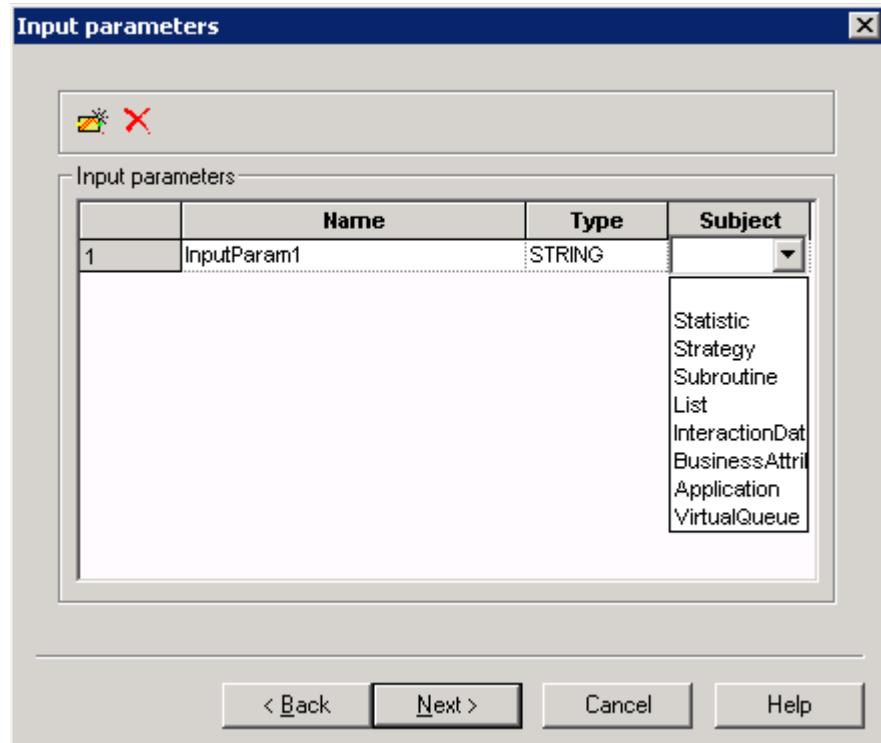


Figure 22: Input Parameters Dialog Box

The dialog box appears after you name and describe the subroutine and click **Next**. You then select the input parameter Type (STRING, FLOAT, or INTEGER) and Subject. Input parameters Subjects are names of types of certain Configuration Server objects. The list currently includes: Statistic, Strategy, Subroutine, List, InteractionData, BusinessAttribute, Application, and VirtualQueue. The Subject of an input parameter functions as hint for the IRD

Application about the possible values that the input parameter can accept. IRD uses this information to provide you with a list of all available Configuration objects of the specified Type when editing an input parameter value.

Data is exchanged between a strategy and subroutine using input and output parameters (see “Configuring Variables” on [page 93](#)) defined in the subroutine and the Call Subroutine object (see [page 121](#)).

When you create a subroutine in IRD, you can create input and output parameters to pass data to and from the subroutine.

- The number of parameters that you create depends on the data you want to pass from the strategy to the beginning of the subroutine and the data you want to pass back to the strategy at the end of the subroutine.
- The number of input parameters does not have to equal the number of output parameters. However, the number of output parameters for a subroutine **must** equal the number of output parameters for the Call Subroutine object.
- Because parameters can be saved in binary format, there is no limitation on the number of characters they can contain.

Warning! If you save an input/output parameter in binary format, you get a warning that earlier IRD versions do not use binary format. This could cause compatibility issues in a mixed IRD environment.

IRD can read both string and binary input and output parameters. Binary format makes it possible to read subroutines with an unlimited number of input/output parameters.

The list of input and output parameters for a strategy is stored in the corresponding `in_param` and `out_param` options of the `Annex` tab in the `Scripts Properties` dialog box in Configuration Manager.

- Parameters created when creating a subroutine are variables and appear in the Subroutine variable list in IRD. (This list also specifies whether the variable is an input or output parameter.)
- Parameters created for the Call Subroutine object (see [page 121](#)) can be variables or constants. Variables used as parameters in a Call Subroutine object appear in the variables list for the strategy. See the *Universal Routing 8.1 Interaction Routing Designer Help* for information on creating variables and variable lists.

After you place a Call Subroutine object in a strategy and specify the subroutine to call, you map the parameters of the Call Subroutine object to the parameters of the subroutine. For example, you can map an input parameter `Variable1` in the Call Subroutine object to an input parameter `SubVariable1` in the subroutine. In this case, the data associated with `Variable1` is passed to the subroutine and becomes associated with

SubVariable1. In a similar way, you map output parameters of the subroutine to output parameters of the Call Subroutine object.

Note: If you are using variables to map parameters between a strategy and a subroutine, make sure the same variable type is defined for both variables. Even though type conversion will occur, type matching is still recommended.

Important Information

- Map all output parameters of a subroutine to an output parameter of the Call Subroutine object. If you do not, an error occurs in the Call Subroutine object.
- After you have changed input or output parameters for a subroutine, you must correct and recompile all strategies where this subroutine is used.
- Select **File > Describe** in the Routing Design window to add, delete, or modify input or output parameters for subroutines and to change their descriptions.

Subroutine Error Codes

Possible error codes for subroutines include the following:

- **0008 Routing done**—This error code is returned if the interaction was routed within the subroutine and no longer needs to continue through the strategy.
- **0018 Unknown object**—This error code is returned if the call subroutine does not exist.
- **0020 Error in subroutine**—This error code is reported to the calling strategy after an interaction reaches an Exit object, if an error occurs within a subroutine. When this error occurs, the interaction is routed using the red output port on the Call Subroutine object. This error does not explain where the error occurs in the subroutine, only that an error occurred.

Note: You can use output parameters not only to pass data but also to pass return codes for error evaluation purposes. A well-developed system of codes defined as parameters can help you resolve subroutine errors. For example, a subroutine contains five function objects. You can set up an output parameter to pass an error that occurs on any of these objects back to the calling strategy.

Routing Rules

Routing rules specify the method of target selection. You can create routing rules once and reuse them in multiple routing objects and strategies.

Many Routing objects (see [page 311](#)) use routing rules in their properties. These include:

- Force (see [page 316](#))
- Load Balancing (see [page 317](#))
- Percentage (see [page 318](#))
- Service Level (see [page 333](#))
- Statistics (see [page 346](#))
- Switch to Strategy (see [page 347](#)).
- Workforce (see [page 353](#))

For example, you might configure a Percentage routing rule that specifies that 82% of interactions go to the San Francisco queue and 18% to the Boston queue. The Routing objects that do *not* use routing rules are the Default (see [page 316](#)), Selection (see [page 326](#)), Route Interaction, Workbin and Queue Interaction objects.

Note: Routing rules are used only by voice routing strategies.

Attributes

Attributes are pieces of interaction or customer data along with all possible values. In order to use a piece of data for strategy decisions, you must first define that piece of data in IRD as an attribute. You must create interaction data before you can create attributes because attributes require interaction data. See “Interaction Data” on [page 72](#).

You can then use the attribute to create business rules. [Figure 23](#) shows example attributes.

Name	Value	Description	Access
Transactions			RHD
alt_account...		account type	RHD
val_existing	InteractionData['account_typ...		
val_new	InteractionData['account_typ...		
alt_language		customer language ...	RHD
val_english	InteractionData['language'] = '1'		
val_spanish	InteractionData['language'] = '2'		

Figure 23: Example Attributes

Create attributes only for data that you plan to segment via business rules (see [page 376](#)).

Note: Do not confuse attribute reusable objects that you define in IRD with Configuration Manager Business Attributes (see [page 98](#)).

Business Rules

Business rules are created from the attributes just discussed (see [Figure 24](#)).

Name	Value	Description	Details	Access
Transactions				RHD
br_existing_english	val_existing	InteractionData['acco...		RHD
br_existing_english	val_english	InteractionData['langu...		
br_existing_spanish	val_existing	InteractionData['acco...		RHD
br_existing_spanish	val_spanish	InteractionData['langu...		
br_new_english	val_new	InteractionData['acco...		RHD
br_new_english	val_spanish	InteractionData['langu...		
br_new_spanish	val_new	InteractionData['acco...		RHD
br_new_spanish	val_spanish	InteractionData['langu...		

Figure 24: Example Business Rules

Business rules and their attributes enable you to create logical expressions for segmentation. They enable you to reuse the same business decision multiple times in the same strategy and in many different strategies.

When you use business rules and attributes, you use a Business Segmentation object (see [page 375](#)) instead of a Generic Segmentation object (see [page 383](#)). The Business Segmentation object properties point to (reference) one or more business rules. Each business rule contains one or more attributes.

For an example of business rule use, see the skills-based routing example in the *Universal Routing Strategy Samples* book. The *Universal Routing 8.1 Interaction Routing Designer Help* contains instructions for creating business rules.

Note: If you make a change to a business rule, you must recompile the strategies that use that rule if you want the change to affect the strategies.

Interaction Data

Interaction data defines attached data keys used in a strategy (see [Figure 25](#)).

Name	Type	Description	Values	M.	M.	Defa...	Access
Transactions							RHD
account_type	string		1,2		1		RHD
language	string	IVR attaches c...	1,2		1		RHD
language_na...	string	language as se...	English		English		RHD
user_account	string	account numb...					RHD

Used in:
Attribute: att_account_type

Uses:

Figure 25: Example Interaction Data

You configure interaction data in order to allow strategies to gather attached data from event messaging, such as for screen pops.

You can use interaction data in the Assign, (see [page 120](#)) and If (see [page 129](#)) objects. You can also use interaction data in certain Function object functions (see [page 127](#)), such as `InteractionData` ([page 461](#)), `InteractionDataINT` ([page 462](#)), `UData` ([page 465](#)), and `BusinessData` ([page 453](#)). In the Function Properties dialog box, select All Functions then Interaction Data.

Schedules

A *schedule* instructs URS when to automatically load/release a voice routing strategy, including the switch and DNs to use. You can specify times for loading and releasing schedules using seconds, minutes, hours, dates, months, years, and days. After you save it, the schedule definition is stored in Configuration Manager as a Script object of type Schedule. See the *Universal Routing 8.1 Interaction Routing Designer Help* for more information.

Notes: When two or more URS instances run in load sharing mode through Load Distribution Server, IRD synchronizes strategy loading so you do not have to load/unload the same strategy on every URS for each routing point in IRD's Monitoring view. This feature applies when you directly load a strategy. If you want to load a strategy via a schedule, it is loaded only to the routing point of the specific URS where you have loaded the schedule. All other URS instances running through the Load Distribution Server in load sharing mode are unaffected.



When URS is processing scheduled items (load/unload), this action is replicated by every instance of URS for its backup server (if one is configured). The secondary (backup) URS does not have to be operational for the loaded strategies to be updated; It is done by the primary URS. The router that is operating in backup mode does not perform any scheduler tasks. The primary URS has the prerogative of performing this action or not

Statistics

In IRD, you can create a custom *statistic* for use in a strategy or subroutine (see Chapter 7, “Routing Statistics,” on [page 715](#) for more information).

List Objects

Note: The IRD installation package supplies sample list objects. For more information, see the Samples chapter in the *Universal Routing 8.1 Deployment Guide*.

A *list object* contains strings of any nature (for example, DNIS or ANI strings), which can be used in strategies and are included in integrity checking.

For example, you may use a list object to create lists of toll-free numbers. Rather than reference each individual 800 number in a strategy, you can logically group numbers together and name the group. Then, when you need to add new numbers or edit numbers, you do not need to edit the strategy, but only the list object properties.

The chapter on Share Agent By Service Level Agreement Routing in the *Universal Routing 8.0 Routing Application Configuration Guide* shows a sample strategy that uses a list object to hold data and uses a special List function (`lcfgdata`) to retrieve the data.

See the *Universal Routing 8.1 Interaction Routing Designer Help* for information on creating list objects.

Note: The ability to define a key-value pair for an element a list object is an extension or variation of the `GetConfigOption` function (see [page 472](#)). Providing key-value pairs for a list element enables you to store information in the Configuration Server Database and then retrieve it from the strategy.

Genesys recommends that you use lists to get user-defined data from Configuration Server. Use a List object that contains all the required data instead of overloading Application options with user-defined data.

Macros

A *macro* is an object that enables you to combine several objects and expressions in one re-usable block. This block works as a user-defined function. See [page 130](#) for more information on macros.

Strategy-Building Objects

IRD provides different categories of strategy-building objects. In toolbar order, the categories are:

- Voice Treatments (see [page 418](#)) to apply busy treatments and mandatory actions, such as a recorded announcement.
- Data and Services (see [page 107](#)) for searching the database for information and sending interactions to external resources.
- Segmentation (see [page 311](#)) for making incoming interactions take different paths.
- Routing (see [page 311](#)) for target selection.
- Miscellaneous (see [page 120](#)) to perform miscellaneous operations.
- Multimedia (see [page 149](#)) to control how non-voice interactions should be routed.
- Outbound (see [Page 294](#)) to support the Genesys Outbound Contact product.
- Workflow and Resource Management (see [Page 392](#)) to enable you to control workflow by changing resource states.
- SMS (see [Page 386](#)) for creating and sending SMS messages.

Icons for each object category appear on the objects toolbar when you create or edit a strategy or subroutine in the Routing Design window (see [Figure 26](#)).

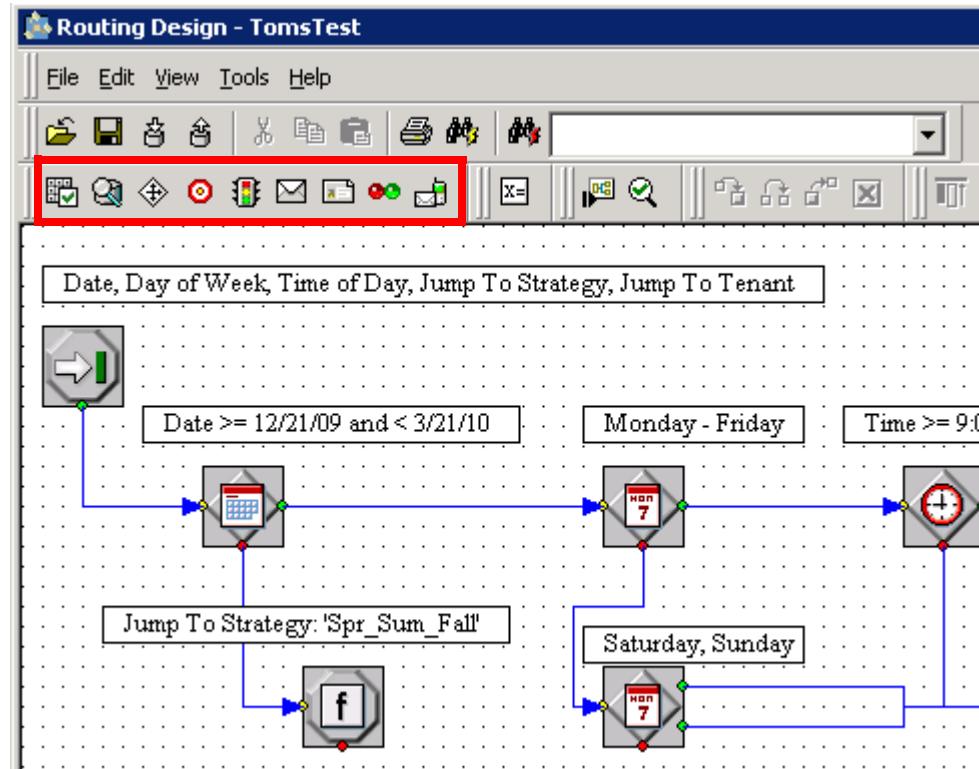


Figure 26: Routing Design Window Toolbar

When you click an icon in the toolbar representing an object category, buttons for all objects belonging to that category drop into view. For example, if you click the third icon, buttons for all Segmentation objects drop into view (see Figure 27).



Figure 27: Drop-Down Buttons for Segmentation Objects

If you were creating a new strategy and wanted to segment incoming interactions to take different paths, you would:

1. Click the applicable Segmentation button (see “Segmentation Objects” on page 418).
2. Click in the strategy to place the object.
3. Double-click the Segmentation object to open a dialog box.
4. Assign properties in the dialog box.
5. Connect the Segmentation object to another object; for example, a Routing object (see page 311) or a Miscellaneous object (see page 120).

A strategy is made up of objects and the connections between objects. All strategies have an Entry object. The other objects you choose define the actions that IRD performs, and the connections that you make between objects define the sequence in which each object is executed. IRD selects only one connection to follow after handling an object.

Toolbar Icons

[Figure 28](#) shows the icons on the toolbar for each strategy-building object category.

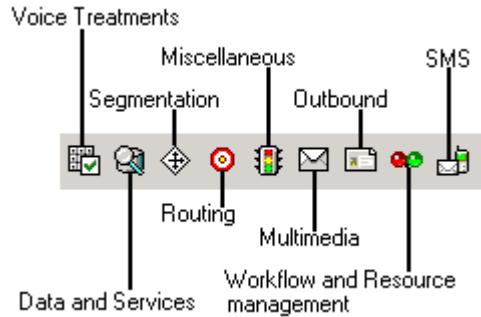


Figure 28: Icons for Strategy-Building Object Categories

The strategy-building objects toolbar is dockable. You can remove it from its current position and place it anywhere within the **Routing Design** window.

Buttons Associated with Toolbar Icons

Clicking an icon on the objects toolbar drops down buttons for placing objects in your strategy. The next section describes the buttons associated with each object category.

Buttons for Voice Treatment Objects

[Figure 29](#) shows the buttons that come into view when you click the Voice Treatments icon.

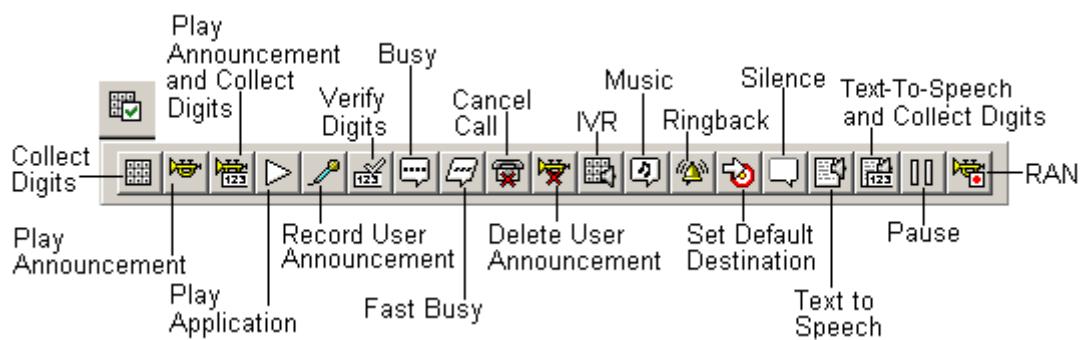


Figure 29: Buttons for Voice Treatment Objects

Voice Treatment objects specify an action to be performed with the current interaction, such as playing music for the caller.

The Voice Treatment objects from left to right are: Collect digits, Play announcement, Play announcement and collect digits, Play application, Record user announcement, Verify digits, Busy, Fast busy, Cancel call, Delete user announcement, IVR, Music, Ringback, Set default destination, Silence, Text to speech, Text to speech and collect digits, Pause, and RAN (Play recorded announcement). See “Access Control” on [page 418](#) for more information.

Buttons for Data and Services

[Figure 30](#) shows the buttons that come into view when you click the Data and Services icon.

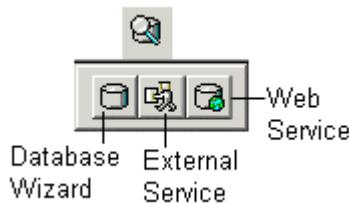


Figure 30: Buttons for Data and Services Objects

The Database Wizard is used for database lookups. You can retrieve data about customers that enable you to personalize responses and route more effectively. See “Data and Services” on [page 107](#) for more information.

Use the External Service object in non-voice routing strategies to exchange data with third party (non-Genesys) servers that use the Genesys Interaction SDK or any other server/application that complies with Interaction Server communication protocol. See “External Service” on [page 115](#) for more information.

You can use the Web Service object to access Web Services. See “Web Service” on [page 119](#) for more information.

Buttons for Segmentation Objects

[Figure 31](#) shows the buttons for various types of Segmentation objects that appear when you click the Segmentation icon.

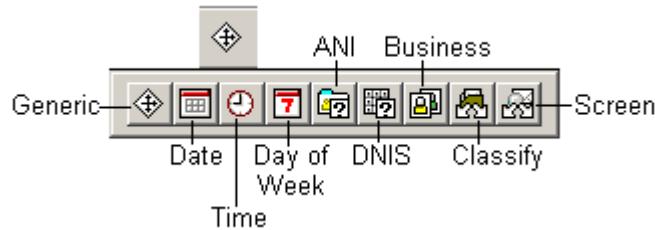


Figure 31: Buttons for Segmentation Objects

Segmentation objects separate incoming interactions and put them on different paths based on criteria that you select.

Note: In most voice strategies, a Segmentation object is the next object after the Entry object.

The buttons from left to right are: Generic (brings up Expression Builder (Figure 39 on [page 88](#))), Date, Time, Day of the Week, ANI (originating phone), DNIS (number dialed), Business, Classify, and Screen. See “Routing Objects” on [page 311](#) for more information.

Buttons for Routing Objects

Click the fourth button on the objects toolbar to reveal the buttons for various types of Routing objects (see [Figure 32](#)).

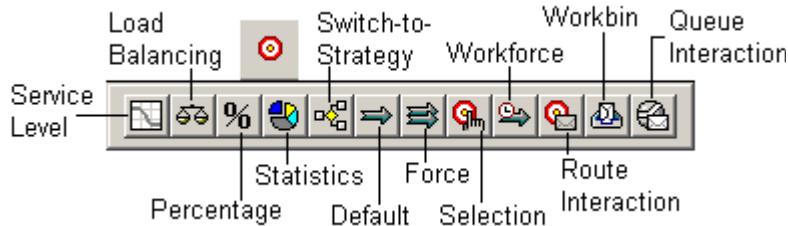


Figure 32: Buttons for Routing Objects

Routing objects specify a routing action to be performed with the current interaction, such as sending an interaction to a specific agent group.

The buttons from left to right are: Service level, Load balancing, Percentage, Statistics, Switch to Strategy, Default, Force, Selection, Workforce, Route Interaction, Workbin, and Queue Interaction. See “Routing Objects” on [page 311](#) for more information.

Buttons for Miscellaneous Objects

Clicking the fifth button on the objects toolbar reveals buttons for various types of Miscellaneous operations (see [Figure 33](#)).

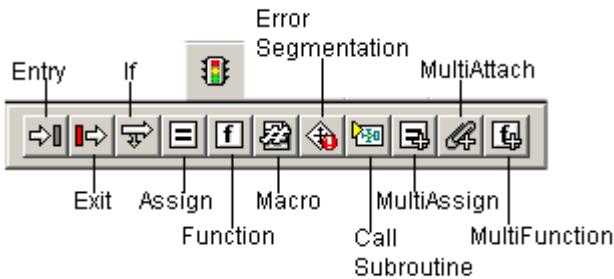


Figure 33: Buttons for Miscellaneous Objects

Miscellaneous objects are used for flow control or performing operations, such as executing a function or directing interactions based on an IF statement.

The buttons from left to right are: Entry, Exit, If, Assign, Function, Macro, Error Segmentation, Call subroutine, MultiAssign, MultiAttach, MultiFunction. See “Miscellaneous Objects” on [page 120](#) for more information.

Buttons for Multimedia Objects

Click the Multimedia icon to reveal buttons for objects used in routing non-voice interactions (see [Figure 34](#)).

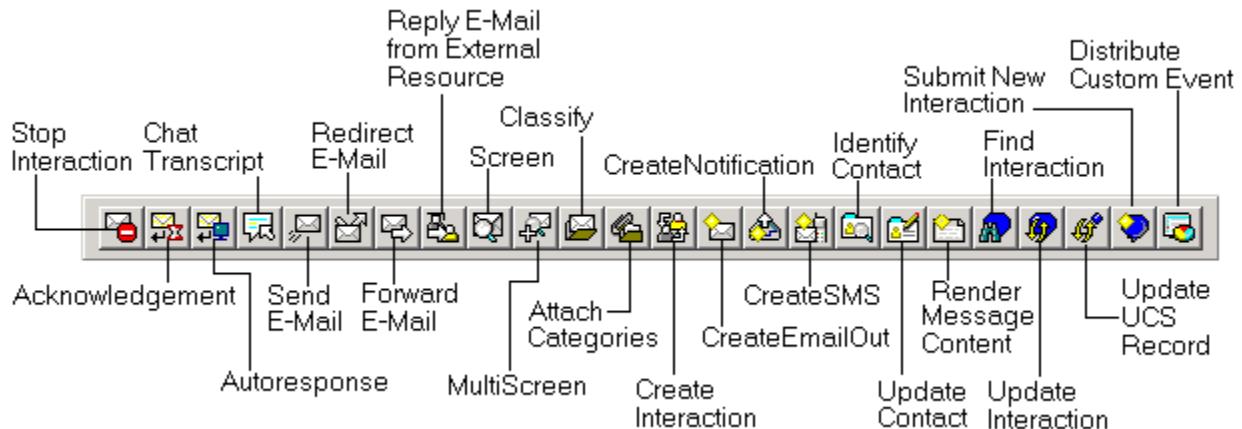


Figure 34: Buttons for Multimedia Objects

The buttons from left to right are: Stop Interaction, Acknowledgement, Autoresponse, Chat Transcript, Send E-Mail, Redirect E-Mail, Forward E-Mail, Reply E-Mail From External Resource, Screen, MultiScreen, Classify, Attach Categories, Create Interaction, Create Email Out, Create Notification, Create SMS, Identify Contact, Update Contact, Render Message Content, Find Interaction, Update Interaction, Update UCS Record, Submit New Interaction, and Distribute Custom Event. See “Multimedia Objects” on [page 149](#) for more information.

Buttons for Outbound Objects

Click the Outbound icon to reveal buttons for objects used to support proactive interaction routing, such as that performed by the Genesys Outbound Contact product. In proactive interaction routing, calls are routed to agents from an Agent/Place Group assigned to the Campaign (see [Figure 35](#)).

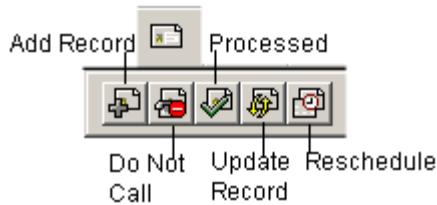


Figure 35: Buttons for Outbound Objects

The buttons from left to right are: Add Record, Do Not Call, Processed, Update Record, Reschedule Record.

Buttons for Workflow and Resource Management Objects

Click the Workflow and Resource Management icon to reveal buttons for objects used to control workflow by changing resource states (see [Figure 36](#)).

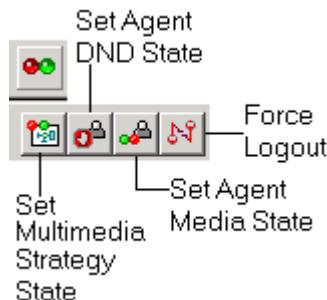
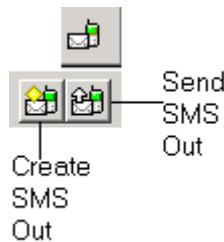


Figure 36: Buttons for Workflow and Resource Management Objects

The buttons from left to right are: Set Multimedia Strategy State, Set Agent DND State, Set Agent Media State, Force Logout.

Buttons for SMS Objects

Click the SMS icon to reveal buttons for objects used for creating and sending SMS messages (see [Figure 37](#)).

**Figure 37: Buttons for SMS Objects**

The buttons from left to right are: Create SMS Out, Send SMS Out.

Comment Object

Use a Comment object to make notes about a strategy. You can insert a Comment object anywhere in the strategy workspace. It appears as a text box in the Routing Design window. For example comments, see Figure 3 on page 29.

Alphabetical List of Strategy-Building Objects

[Table 1](#) lists all available strategy-building objects in alphabetical order, the category or group under which the object is placed in IRD, and the page number for the object information in this chapter.

Table 1: Strategy-Building Objects

Object	Type	Page
Acknowledgement	Multimedia	161
Add Record	Outbound	295
ANI	Segmentation	374
Assign	Miscellaneous	120
Attach Categories	Multimedia	178
Autoresponse	Multimedia	181
Business	Segmentation	418
Busy	Voice Treatments	407
Call Subroutine	Miscellaneous	121
Cancel Call	Voice Treatments	407
Chat Transcript	Multimedia	191

Table 1: Strategy-Building Objects (Continued)

Object	Type	Page
Classify	Multimedia	198
Classify (segmentation)	Segmentation	377
Collect Digits	Voice Treatments	407
Comment	None	81
Create Interaction	Multimedia	205
Create Email Out	Multimedia	212
Create Notification	Multimedia	217
Create SMS	Multimedia	221
Create SMS Out	SMS	386
Database Wizard	Data and Services	107
Date	Segmentation	380
Day of Week	Segmentation	381
Default	Routing	316
Delete User Announcement	Voice Treatments	409
Distribute Custom Event	Multimedia	226
DNIS	Segmentation	381
Do Not Call	Outbound	299
Entry	Miscellaneous	126
Error Segmentation	Miscellaneous	123
Exit	Miscellaneous	127
External Service	Data and Services	115
Fast Busy	Voice Treatments	409
Find Interaction	Multimedia	228
Force Logout	Workflow and Resource Management	393
Force Routing	Routing	316

Table 1: Strategy-Building Objects (Continued)

Object	Type	Page
Forward E-Mail	Multimedia	233
Function	Miscellaneous	127
Generic	Segmentation	383
Identify Contact	Multimedia	237
If	Miscellaneous	129
IVR	Voice Treatments	409
Load Balancing	Routing	346
Macro	Miscellaneous	130
MultiAssign	Miscellaneous	140
MultiAttach	Miscellaneous	140
MultiScreen	Multimedia	241
Music	Voice Treatments	410
Pause	Voice Treatments	411
Percentage	Routing	318
Play Announcement	Voice Treatments	411
Play Announcement and collect digits	Voice Treatments	412
Play Application	Voice Treatments	412
Processed	Outbound	302
Queue Interaction	Routing	320
RAN	Voice Treatments	413
Record User Announcement	Voice Treatments	414
Redirect E-Mail	Multimedia	250
Render Message Content	Multimedia	254
Reply E-Mail From External Resource	Multimedia	258
Reschedule	Outbound	305

Table 1: Strategy-Building Objects (Continued)

Object	Type	Page
Ringback	Voice Treatments	415
Route Interaction	Routing	322
Screen	Multimedia	263
Screen	Segmentation	384
Selection (target selection)	Routing	326
Send E-Mail	Multimedia	269
Send SMS Out	SMS	390
Service level	Routing	333
Set Agent DND State	Workflow and Resource Management	395
Set Agent Media State	Workflow and Resource Management	398
Set Default Destination	Voice Treatments	415
Set Multimedia Strategy State	Workflow and Resource Management	400
Silence	Voice Treatments	415
Statistics	Routing	346
Stop Interaction	Multimedia	274
Submit New Interaction	Multimedia	277
Switch to Strategy	Routing	347
Text to Speech	Voice Treatments	415
Text to Speech and Collect Digits	Voice Treatments	416
Time	Segmentation	385
Update Contact	Multimedia	281
Update Interaction	Multimedia	285
Update Record	Outbound	307
Update UCS Record	Multimedia	290

Table 1: Strategy-Building Objects (Continued)

Object	Type	Page
Verify Digits	Voice Treatments	416
Web Service	Data and Services	771
Workbin	Routing	348
Workforce	Routing	353

The later sections of this chapter describe the objects and their properties in detail. Functions associated with the Function object are described in Chapter 4, “Interaction Routing Designer Functions,” on [page 423](#).

Object Properties Dialog Boxes

Every IRD object has a set of properties or parameters that describe a portion of the strategy. The exceptions to this are: the Entry, Exit, and Default Routing objects; the Cancel Call Voice Treatment object; and a few Function objects.

Double-click a strategy-building object in the `Routing Design` window workspace to open its properties dialog box. Here you can assign values to the object’s parameters. The content of the properties dialog box varies based on the object.

[Figure 38](#) shows an example properties dialog box for the Selection object.

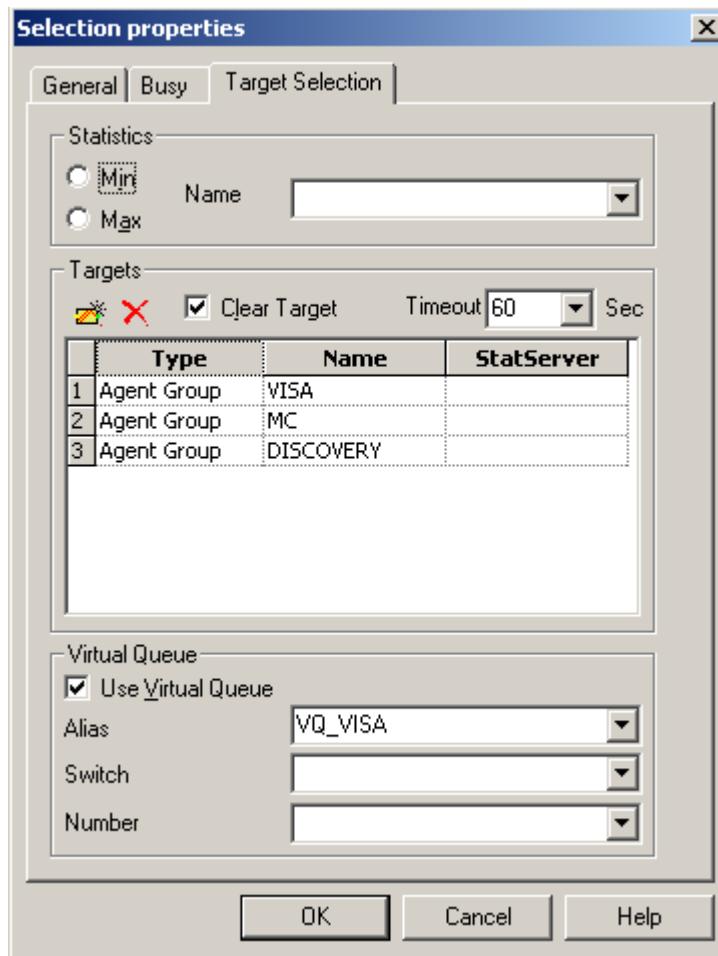


Figure 38: Selection Properties Dialog Box

The pages ahead show example properties dialog boxes for strategy-building objects in each object category.

The parameters of each object in a properties dialog box are specific to its category.

For example:

- In the Selection properties dialog box shown in [Figure 38](#), the parameters you specify are routing targets (see Table 112, “Target Types,” on [page 359](#)). A routing target can also be specified as a skill expression.
- The parameters of some Segmentation objects are logical expressions (see [Figure 39 on page 88](#)).
- The properties dialog box for the Function object includes the function name, parameters and values (see [Figure 66 on page 128](#)).
- The parameters of Voice Treatment objects are the type of treatment and some supplemental data. Voice Treatments can also be used as parameters of Routing objects.

Both parameters and voice treatments are ways in which URS handles an interaction. Parameters are criteria defining how the interaction is handled, such as how long it should be held before being sent to a default extension, its priority in queue, and so forth. Voice treatments, by contrast, affect what the caller hears, for example, music, silence, or a busy signal.

Important Information

- Deleting or changing the name of any item (rules, objects, targets, statistics, and so on) that is referred to in other strategies, rules, objects, or functions invalidates the strategy unless the object using the referenced item is updated. (Before changing or deleting any object, go to the List view for the object to see which strategies will be affected and then modify them as needed so they are not rendered invalid.)

Use the Check Integrity tool to validate all strategies and confirm the existence of all referenced statistics. See the *Universal Routing 8.1 Interaction Routing Designer Help* for instructions on how to use this tool.

- When using string constants in a strategy, the number of characters that make up the constant (what appears between the quotes) is limited to 1000. For example, Assign[x, '...over 1000 characters'] is not supported.
- A string constant used in a strategy cannot contain quotes.

Logical Expressions

The properties of several IRD objects are defined as logical expressions. These are created using:

- The Expression Builder in the Generic Segmentation object (see [Figure 39](#)).
- The Skill Expression Properties dialog box of a Routing object (see [page 90](#)).
- The If object (see [Figure 67 on page 130](#)).
- The Assign object (see [Figure 61 on page 120](#)).
- Some Function objects (see [page 89](#)).

Expression Builder

[Figure 39](#) shows the Expression Builder dialog box after you have constructed an expression.

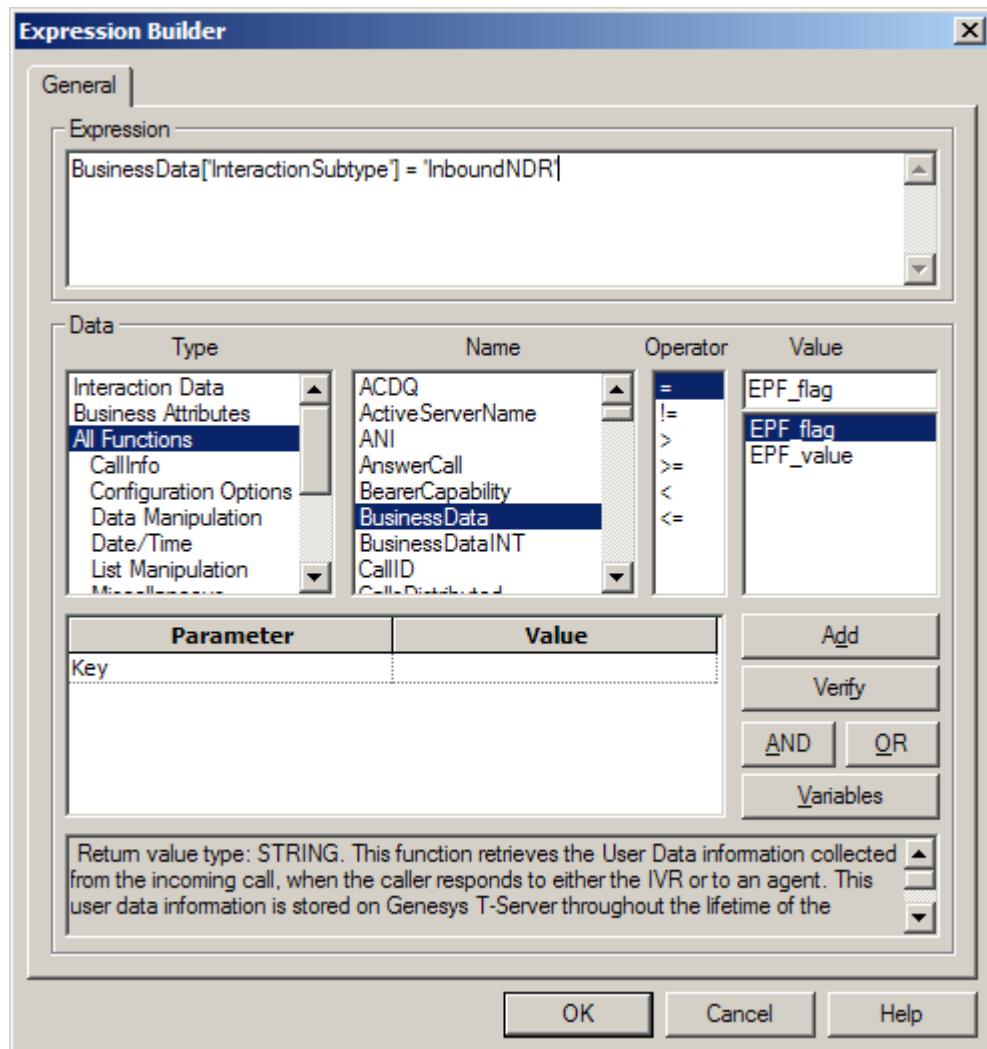


Figure 39: Expression Builder Dialog Box

Skill Expressions

Note: Also see “Skill Target in Routing Objects” on [page 363](#) and “Skill Targets and Skill Expressions” on [page 365](#),

You can route interactions to the most appropriately skilled agent using a skill expression. [Figure 40](#) shows the Skill Expression Properties dialog box after constructing a skill expression for the Routing Selection object.

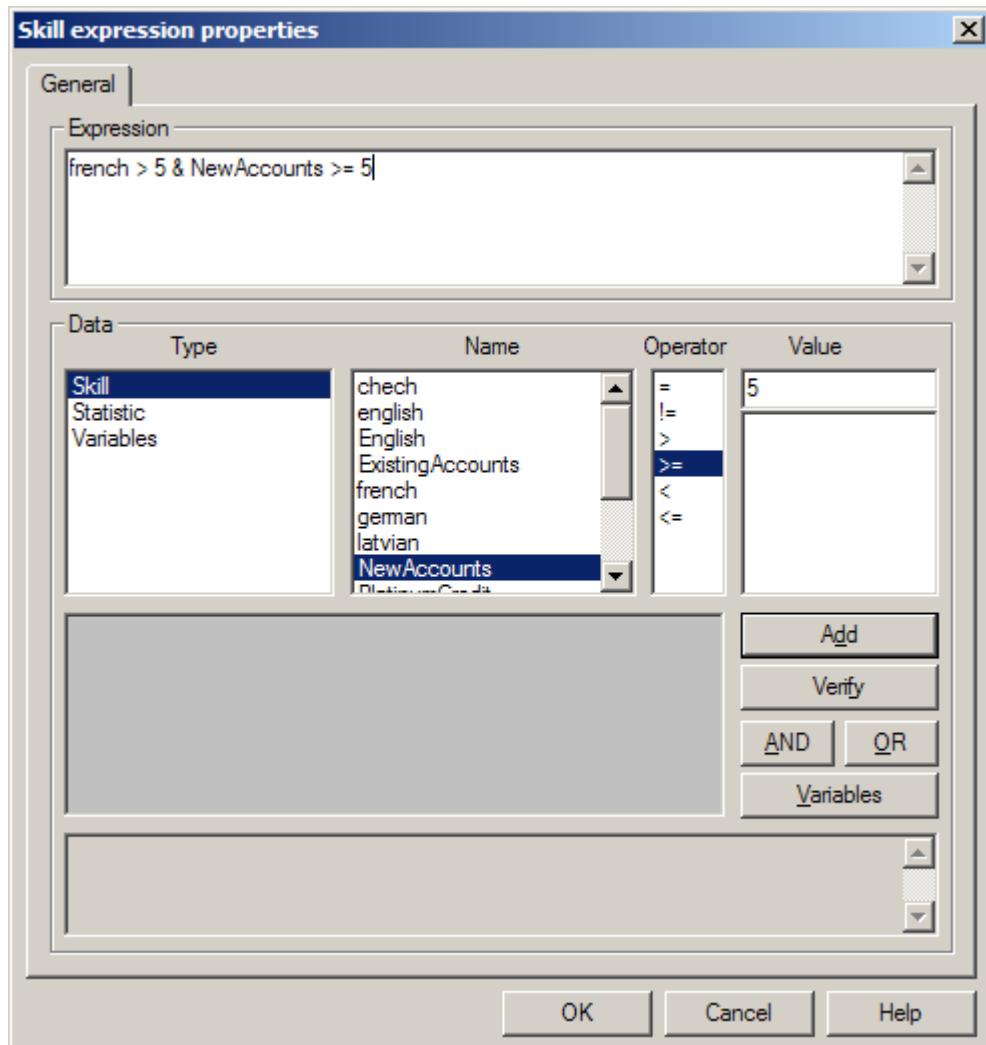


Figure 40: Skill Expression Properties Dialog Box

Note: Stat Server currently supports virtual group functionality for the following types of agent parameters: a `Skill` configured for an agent, the `ACDQueue` to which an agent is logged in, and the `switch` to which an agent is logged in. You can simultaneously specify these types of parameters in an expression for a single virtual group. For more information, see the Virtual Agent Groups chapter in the *Framework 8.1 Stat Server User's Guide*.

- Functions that support skill expressions include `CountSkillInGroupEx` (see [page 511](#)), `GetSkillInGroupEx` (see [page 519](#)), `CreateSkillGroup` (see [page 514](#)), and `Multiskill` (see [page 526](#)).

- These Routing objects allow skill-routing targets: Selection ([page 326](#)), Statistics ([page 346](#)), Service Level ([page 333](#)), Route Interaction ([page 322](#)), and Workbin (see [page 348](#)). See [page 363](#) for more information on using skills as routing targets.

The name of skills, if used in a skill expression, cannot exceed 126 bytes.

Warning! If configuring skill or agent level routing for a site with a large number of agents, Genesys recommends using the StatAgentLoading ([see page 722](#)) or StatAgentLoadingMedia ([page 724](#)) statistic to select agents. Using the StatTimeInReadyState statistic can result in high CPU usage even if there are no calls.

Building Expressions

Logical expressions consist of comparisons joined by the connectives AND (&) and OR (|).

Notes: The AND and OR logical connectives have the same priority.

URS uses integer arithmetic in its calculations, such as for agent state and skill expression evaluation. For this reason, you must always create expressions based on integer arithmetic, not float.

When an expression contains more than one logical connective, the logical operation to the left has precedence over the operation to the right; use parentheses to overrule this precedence.

Each comparison consists of two data values that are compared against each other. [Table 2](#) shows the comparison symbols used in logical expressions. The left side of the comparison specifies the type of information provided on the right side (the comparable). For example, Day=07/04/01 and Time>=18:00 are comparables for Day and Time. The left side can be interaction data ([page 72](#)), a variable ([page 92](#)), or a function (Figure 39 on [page 88](#)). However, the choices vary depending on the type of object being defined.

Table 2: Comparison Symbols in Logical Expressions

Symbol	Meaning	Example
=	equal to	Day = Monday
!=	not equal to	Day != Sunday
>	greater than	Time > 9:00
>=	greater than or equal to	Day >= Monday

Table 2: Comparison Symbols in Logical Expressions (Continued)

Symbol	Meaning	Example
<	less than	Time < 16:00
<=	less than or equal to	Day <= Friday

[Table 3](#) provides several examples of logical expressions.

Table 3: Examples of Logical Expressions

Example	Interpretation
Day=Saturday Day=Sunday	If the day is Saturday or Sunday
Time<=5:00 Time>=18:00	If the time is less than or equal to 5:00 (5:00 AM) or greater than or equal to 18:00 (6:00 PM); in other words, if the time is between 6:00 PM and 5:00 AM.
Day>=Monday&Day<=Friday & ANI!=8004154732	If the day is a weekday and the ANI is not 8004154732
SData[1123@SF.A,StatTimeInReadyState]>120	If agent 1123 has been in Ready state for more than 120 seconds
UData[Acct]>=9000	If the value associated with the key Acct in the Interaction data structure is equal to or greater than 9000

Mathematical Symbols Allowed

The allowed mathematical symbols in expressions are restricted to those that IRD allows in the Assign object (see [page 120](#)). These are the mathematical symbols for addition, subtraction, division, and multiplication (+, -, /, *).

These symbols work with numeric values only, so any participating String argument is automatically converted to a number.

Variables in Objects and Expressions

Warning! Variables are specific to the strategy or subroutine in which the variable was defined and cannot be used by other subroutines or strategies, including the strategy that calls the subroutine.

Variables enable you to specify such things as targets, servers, parameter values for functions, skill expressions, mandatory treatments, and so on greater flexibility. For example, in combination with the Assign object (see [page 120](#)) you can route interactions based on a value assigned to that variable.

Variable Usage and Strategy Design

Note: You cannot define variables for routing rules, since routing rules can be used by more than one strategy.

Variables are particularly useful for avoiding multiple queries to a database for a particular interaction. One database lookup can obtain all necessary customer information that can be assigned to different variables (see Figure 55 on [page 111](#)). The information can then be used to route the interaction. Only one lookup is needed because of variable usage. This makes the strategy is more efficient and improves software performance.

Using variables you can also create a generic strategy where segmentation and routing targets are driven by variables. This generic strategy can contain all the business rules (see [page 71](#)) needed to make the routing decision with the routing target specified as a variable. Then you can load this generic strategy to multiple routing points and have routing decisions and targets decided at runtime. This replaces the need to create a different strategy for each routing point, each with its own routing requirements.

Note: Variable-driven strategies are an advanced feature in IRD and require a user with a programming background to properly design.

Variables are specific to the strategy in which they are created. As such, variables cannot be used in routing rules or by any object that can be used by more than one strategy. The only exception to this is a subroutine object. Variables can be defined for a subroutine, which then can be used by other strategies. However, these variables can only be used in the subroutine.

Configuring Variables

Variables must be configured before you can use them. Use the Variable List dialog box for this purpose. [Figure 41](#) shows the Variable List dialog box with six variables defined.

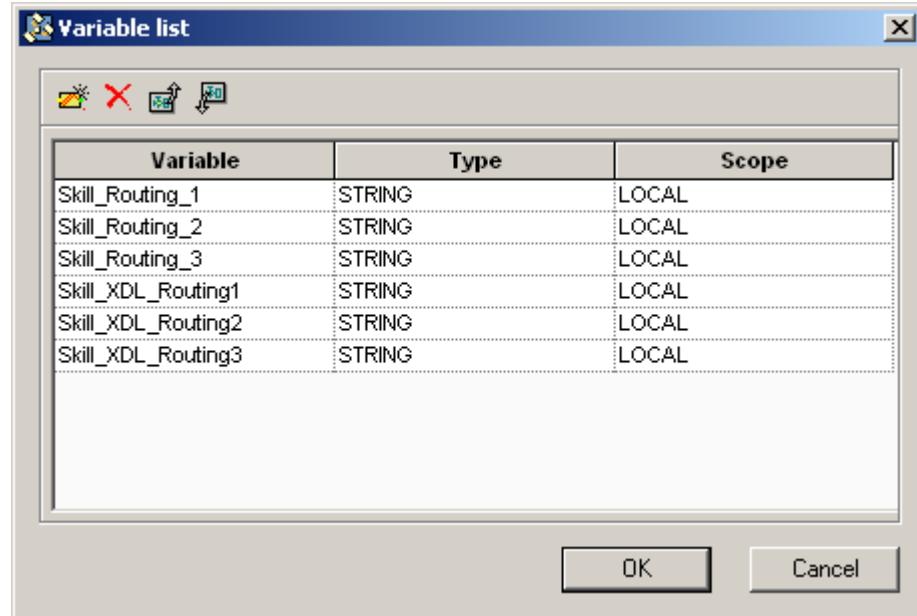


Figure 41: Variable List Dialog Box

Once variables are defined, you can use them for routing in the TargetSelection tab of both the Selection object (see [page 326](#)) and Route Interaction object (see [page 322](#)). [Figure 42](#) shows the above variables available in the Selection object when routing voice interactions.

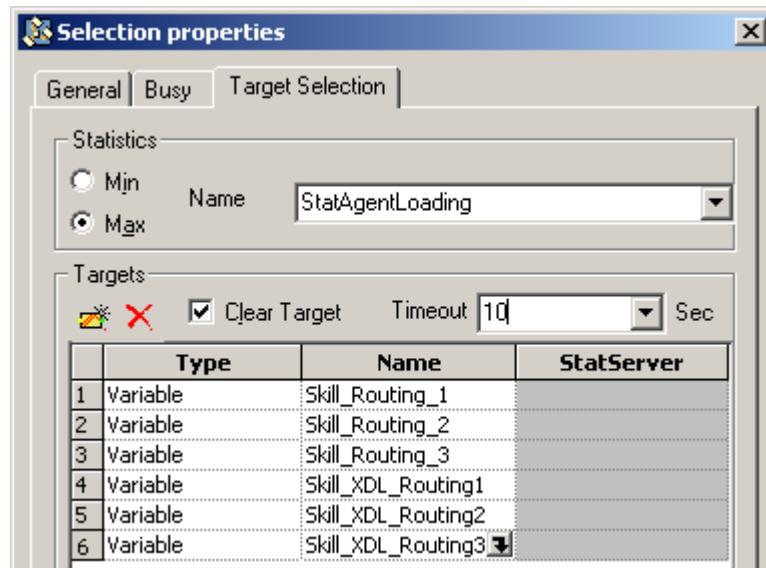


Figure 42: Routing Based on the Value of a Variable

To open the Variable List dialog box:

- When working with a strategy, click the variables button  on the toolbar to bring up this dialog box. The fields for defining a variable include variable name, type, and scope.
- In the Database Wizard (see [page 108](#)), when you assign the output or key-value pairs to one or more variables, a dialog box opens (see Figure 56 on [page 111](#)) where you click the Edit Variables button to bring up the Variable List dialog box.
- In the Generic Segmentation object (see [page 383](#)), click the down arrow in the Expression column to bring up the Expression Builder (see Figure 201 on [page 383](#)). In the Expression Builder, click the Variables button.
- In the properties dialog box for the If (see [page 129](#)), Assign (see [page 120](#)), or Function (see [page 127](#)) Miscellaneous objects, click the Variables button to bring up the Variable List dialog box.
- The Multi-Assign object (see Figure 76 on [page 140](#)) lets you assign multiple variables at once using its own dialog box.

For more information about variable formats, see [Table 4](#) and [Table 5](#) on [page 97](#), and also [Table 6](#) on [page 97](#).

Assigning Values to Variables

After you configure a variable in the Variable List dialog box, you can assign a value to it. Use one of the following objects to assign a value:

- Assign object, in the Expression dialog box that opens from this object (see Figure 61 on [page 120](#))
- Database Wizard (see Figure 55 on [page 111](#))
- MultiAssign object (see Figure 76 on [page 140](#))
- Function object, in the Expression dialog box that opens from this object.

Assigning a Variable in the Function Object

[Figure 43](#), under Expression at the top left, shows the output of the Rand function assigned to the variable `min_skill_level`.

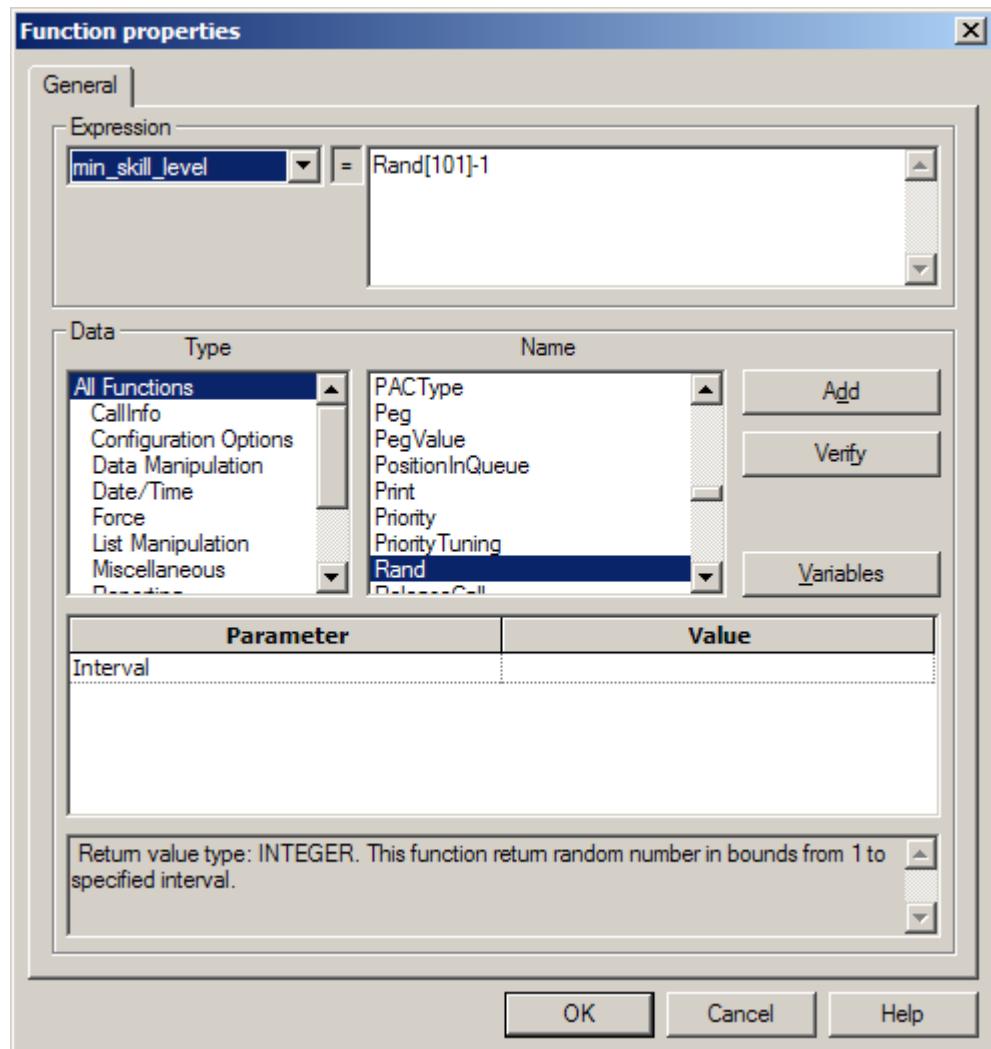
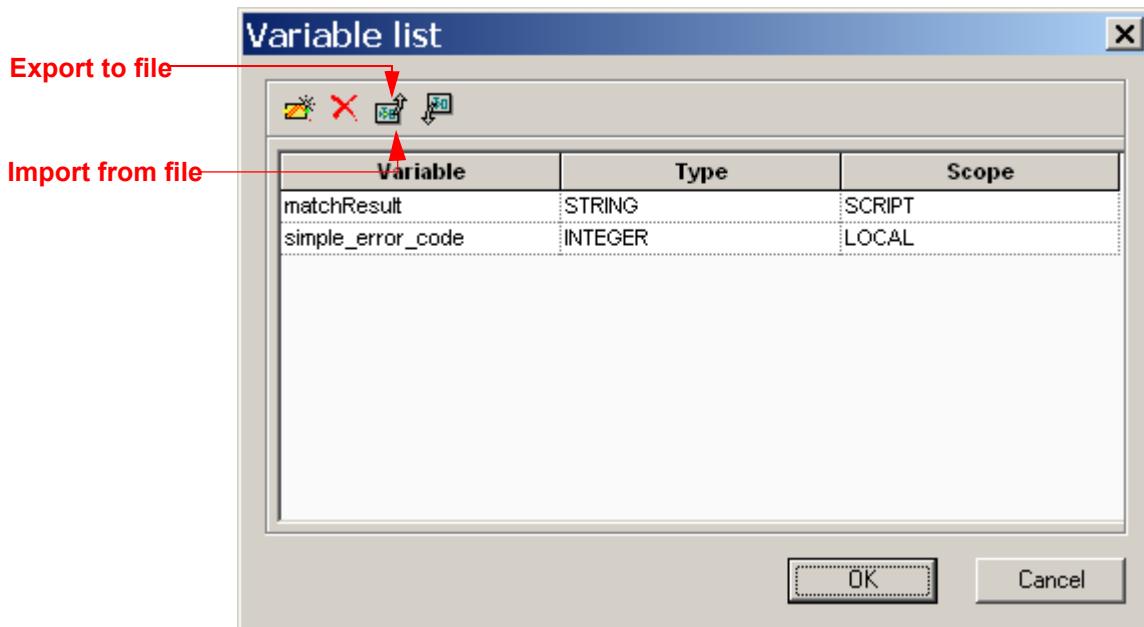


Figure 43: Function Object Properties Dialog Box

You can assign an explicit value (1234). Or you can use other variables as values to assign to a variable.

Exporting and Importing Variables

You can move definitions of variables used in one strategy to another strategy using the Export and Import buttons on the Variable List dialog box (see Figure 44).

**Figure 44: Variable List Dialog Box, Export and Import Buttons**

- Clicking the `Export to file` button opens a dialog box where you can save the variables to file for later import.
- Clicking the `Import from file` button opens a dialog box where you can import variables previously saved to a file.

Specifying Variables

[Table 4](#) and [Table 5](#) below contain information on specifying variables.

Table 4: Variable Specifications

Variable Specification	Explanation
Name	Can be mixed case. No spaces allowed. Cannot include mathematical symbols, hyphens, quotes, or double-byte characters (international only) or be a function name. Variable names also cannot begin with BL (case-sensitive). BL is an internal identifier used by IRD.
Quotes	IRD checks every field where both variables and constants can be entered. A string (and only it) will <i>not</i> be enclosed in single quotes if at least one is true: It is a variable name It is already included in single quotes. It has '[' inside and doesn't start with it (nested function call).

Table 4: Variable Specifications (Continued)

Variable Specification	Explanation
Definition	A value
Limitations	A variable name cannot exceed 64 characters.
Example of Definition	VARIABLE INTEGER LOCAL Product
Example of Usage	Assign[Product,UData[prod]] Product = 1 ->

Table 5: Variable Types

Type	Description	Examples
STRING	A case-sensitive text string.	CA 2
INTEGER	A 32-bit signed integer.	1, 5, 100
FLOAT	A floating number.	1.2, 3.54
LIST	List of values of any valid type: STRING, INTEGER, FLOAT, or LIST. List members are accessed by keys.	user_data = {FirstName= "Rodolphe", ContactId = "0005Ub6CJC6H0001", LastName = "Godreul"} Name:Greg Lastname:Smith phone:19253571243

Note: When creating variables, IRD remembers the type of the last variable created. This type becomes the default until you create a new variable with a different type.

Table 6: Variable Scope

Scope	The variable retains its value
LOCAL	For the life of one particular interaction currently being processed in this script (strategy) only. This value is re-created upon the arrival of each new interaction.
SCRIPT	For all interactions within one script (strategy).

Table 6: Variable Scope (Continued)

Scope	The variable retains its value
USER	A USER variable is created when the tenant is active, and it is destroyed when the tenant is released from URS's memory. Everything running within the tenant can change the value of the same USER variable. Values are shared within the current tenant.
GLOBAL	A GLOBAL variable is created when the particular instance of URS runs, and it is destroyed when the URS stops running or is released from memory. Everything running on this instance of URS can change the value of the same GLOBAL variable. Values are shared within the entire URS instance.
INTERACTION	An INTERACTION variable is created when a particular interaction is active, and it is destroyed when the interaction ends. The value of an INTERACTION variable for one interaction has no effect on the value of the same INTERACTION variable for another interaction. Values are shared for the current interaction only.

The Interaction Design Window

In addition to the Routing Design window for creating routing strategies, IRD has an Interaction Design window for creating and editing *business processes* (see Figure 46 on [page 100](#)).

Business Processes

A business process directs non-voice customer interactions arriving at the contact center through various processing objects, such as queues and routing strategies. It defines what happens to customer interactions (such as e-mail, chat, and open media) from the point of arrival to the point of completion. An *interaction workflow* comprises one or more business processes.

The types of processing applied to interactions varies based on the media type and the contact center's business logic. In most cases, the goal is to generate an appropriate response for the customer.

- In the case of an e-mail interaction, an appropriate response might be an e-mail answering the customer's questions.
- In the case of a chat interaction, an appropriate response might be mailing product brochures to the customer.
- In the case of a fax interaction, an appropriate response might be an e-mail stating the requested materials had been received, and so on.

Opening Interaction Design

In the IRD main window, click the **Interaction Design** shortcut bar, and then click the **Business processes** icon. Existing business processes appear in the **Business processes** list (see [Figure 45](#)).

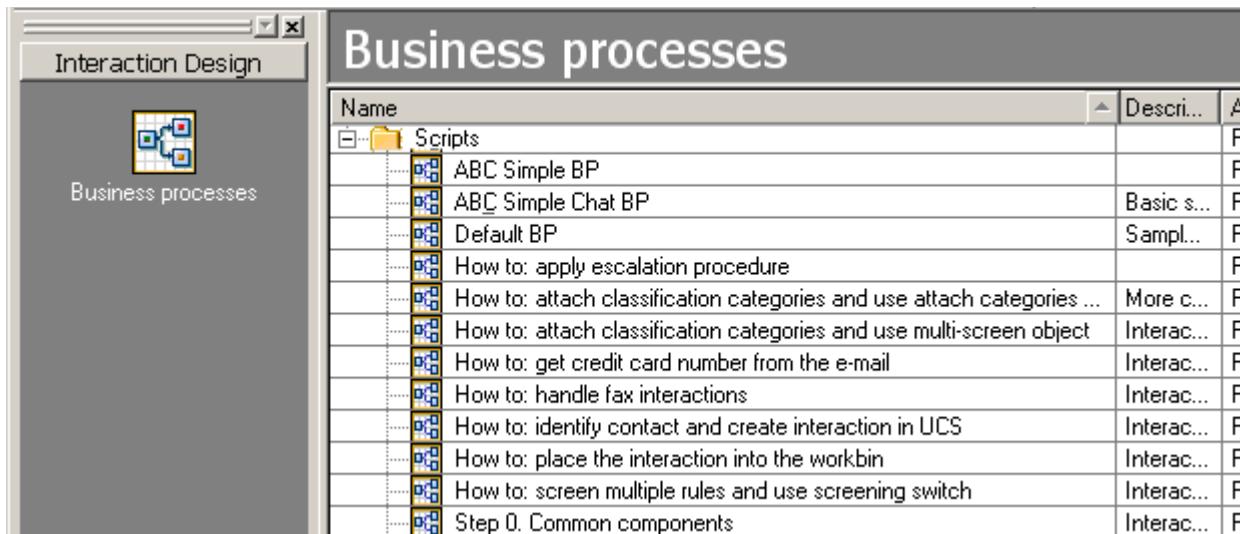


Figure 45: Business Processes List

Note: By default, the **Interaction Design** shortcut bar only appears if Genesys eServices components were installed and configured in your environment by the eServices Configuration Wizard. As a result of this installation, an eServices solution should appear in the **Solutions** folder under **Environment**. If you manually install Genesys eServices, the **Interaction Design** bar does not appear. In this case, overwrite default IRD behavior in the following way:

1. After installation of IRD completes, open IRD and select **Routing Design Options** from the **Tools** menu.
2. Click the **Views** tab and uncheck the **Default** check box.
3. Verify that following check boxes are checked: **Interaction Design**, **Business processes**.
4. Click **OK** to save your changes.

Creating a New Business Process

From the **Business processes** list, click the **New** icon or select **File > New** from the IRD main window menu. To edit an existing business process, double-click it in the **Business processes** list. The business process opens in the **Interaction Design** window, as shown in [Figure 46](#).

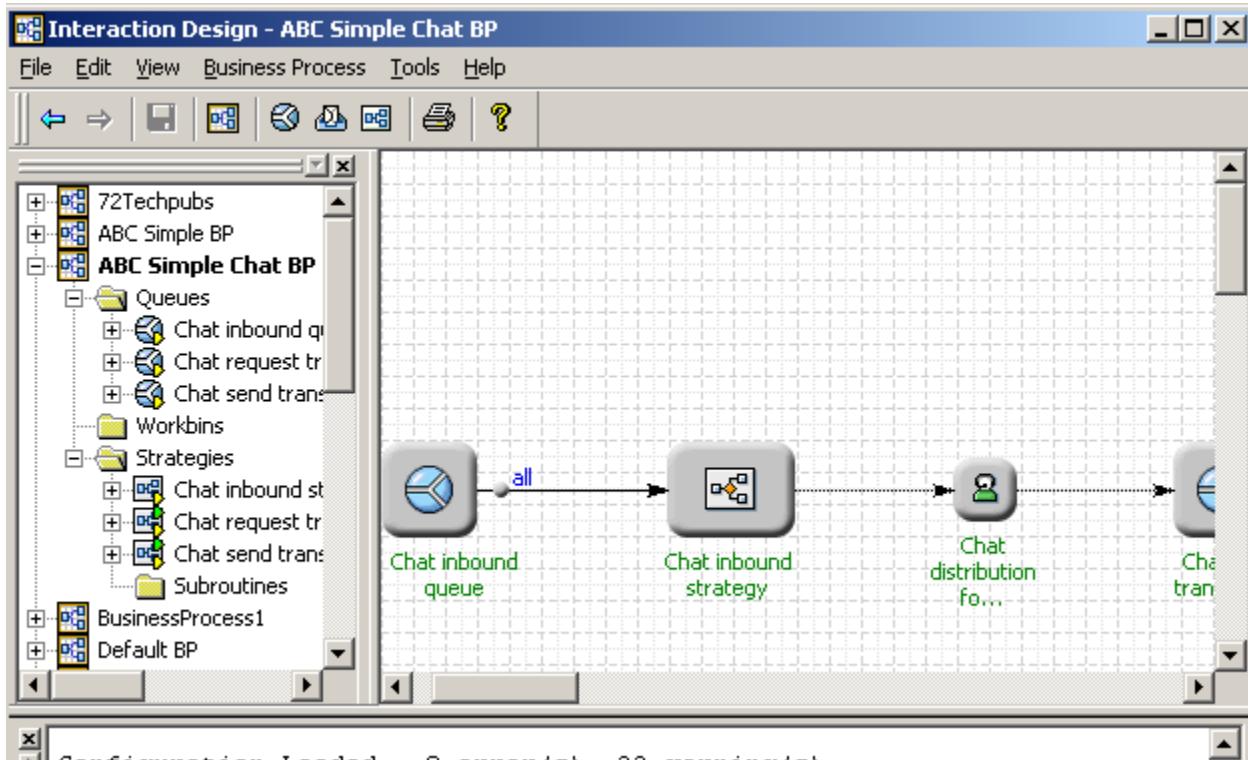


Figure 46: Interaction Design Window

As can be seen in [Figure 46](#), the Interaction Design window has three areas:

1. An object browser (dockable) on the left that displays business processes, queues, workbins, and routing strategies in a tree style.
2. A workflow viewer (static) on the right that graphically displays interaction flow for selected business process.

Note that the layout of Business Processes (arrangement of Business process blocks) can be changed in either of two ways:

- explicitly by a user when editing the Business Process.
- implicitly by the Interaction Design editor. When one of the strategies in the Business Process is updated by adding new targets or certain multimedia servers, the Interaction Design editor will rearrange all objects in accordance with its internal logic. In this case, the customized arrangement of the Business Process objects in Workflow Viewer is not preserved. For more information on objects/settings that could cause involuntary rearrangement of Business Process layout please refer to the Workflow Settings Tab Options topic of Interaction Routing Designer Help.

3. A log viewer (dockable) with three tabs (Log, Configuration Updates, Search Results) that display various lists of status and event messages.

For more information, see the *Universal Routing Business Process User's Guide*.

Associating a Business Process with Accounts

As an optional last step, you may consider associating a Business Process with Accounts.

Accounts can be assigned to different Interactions servers (refer to the Restricted release of the *eServices Tools User's Guide* for additional information). This then results in Business Processes from specific accounts to be processed only with the corresponding Interaction Servers.

To create an Account:

From the Interaction Design window:

Select the Business Process menu

Then select New Account (See [Figure 47](#))

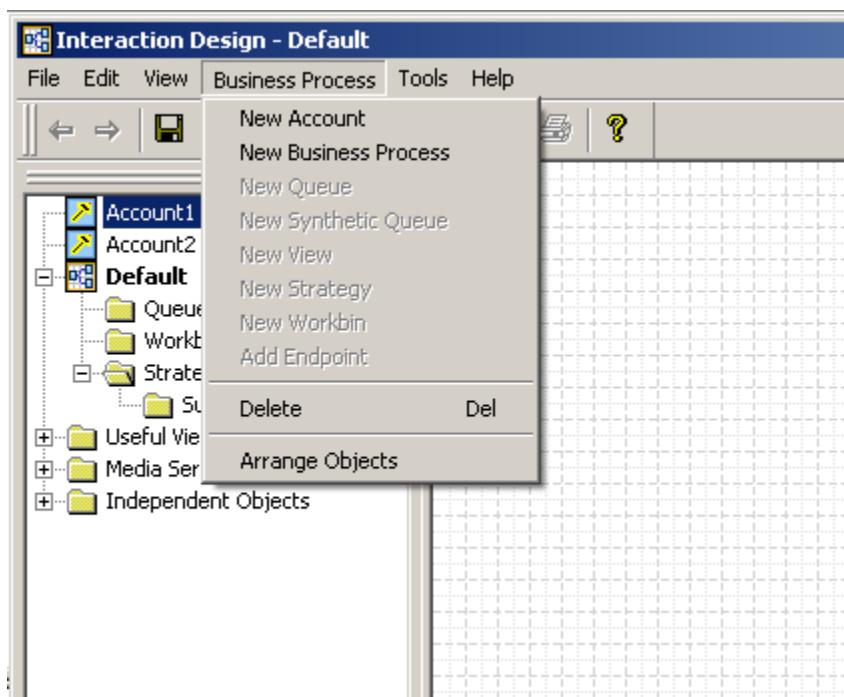


Figure 47: New Account

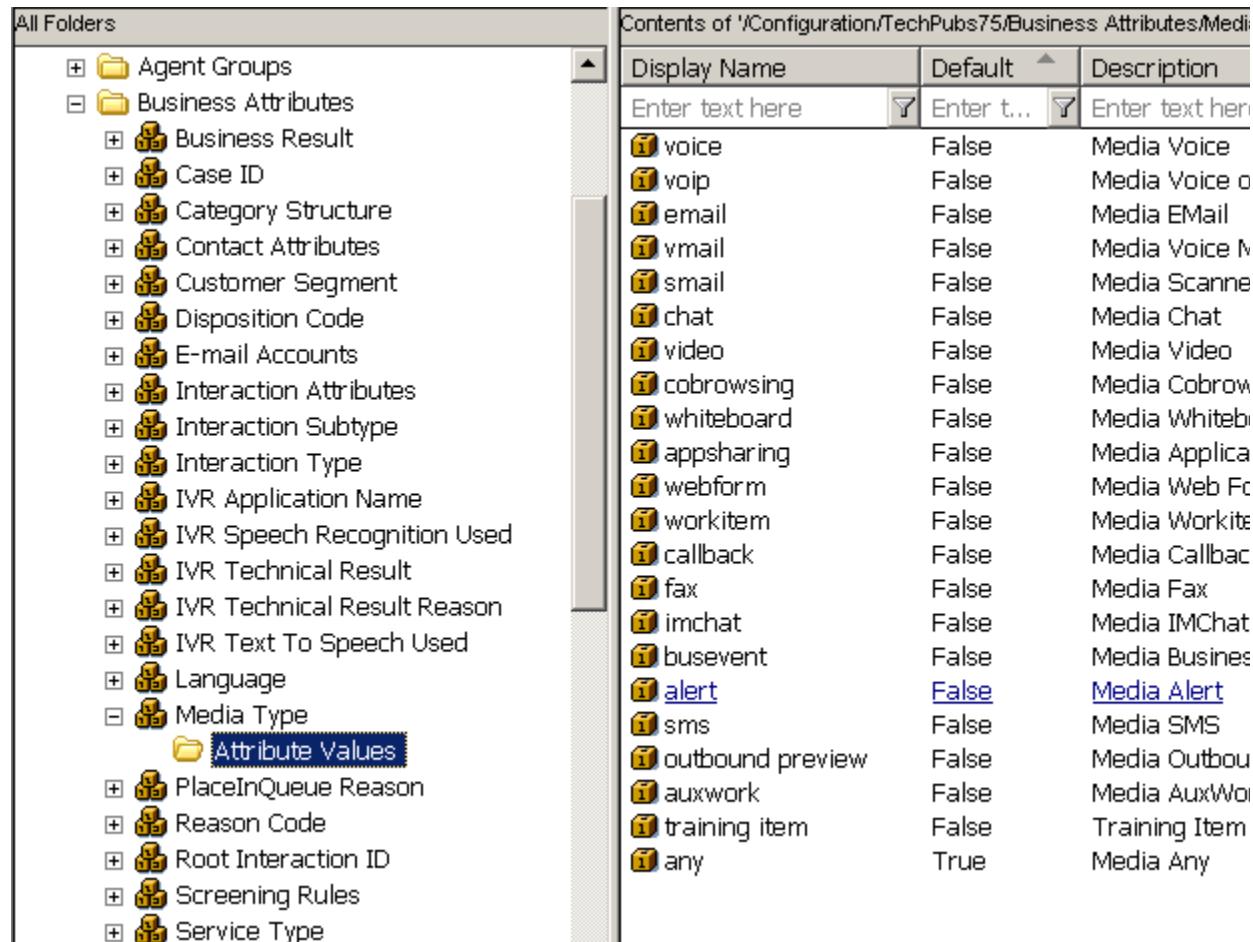
To associate or re-associate a Business Process with the Account:

- Drag and drop the Business process object to or from the Account.

See the Creating a Business Process section of the *Universal Routing 8.1 Interaction Routing Designer Help* for additional information.

Business Attributes

Configuration Manager contains both predefined and user-created Business Attributes (see [Figure 48](#)).



The screenshot shows the Configuration Manager interface with the 'Business Attributes' folder selected in the left navigation tree. The right pane displays a table of business attributes with columns for Display Name, Default, and Description.

Display Name	Default	Description
voice	False	Media Voice
voip	False	Media VoIP
email	False	Media Email
vmail	False	Media Voice Mail
smail	False	Media Scanner
chat	False	Media Chat
video	False	Media Video
cobrowsing	False	Media Cobrowsing
whiteboard	False	Media Whiteboard
appsharing	False	Media Application Sharing
webform	False	Media Web Form
workitem	False	Media Workitem
callback	False	Media Callback
fax	False	Media Fax
imchat	False	Media IM Chat
busevent	False	Media Business Event
alert	False	Media Alert
sms	False	Media SMS
outbound preview	False	Media Outbound Preview
auxwork	False	Media Auxiliary Work
training item	False	Training Item
any	True	Media Any

Figure 48: Configuration Manager Business Attributes

Business Attributes are interaction attributes that are used in different ways by various Genesys applications and by strategy building objects contained within routing strategies used by business processes (Figure 83 on [page 149](#) shows an example business process).

The origin of *Business Attributes* varies:

- Some *Business Attributes* are predefined by Genesys and can be expanded.
- Other *Business Attributes* are customer-defined in Configuration Manager.
- Other *Business Attributes* are initially defined in Knowledge Manager and then carried over to Configuration Manager.

Note: See the *Genesys eServices (Multimedia) 8.1 User's Guide* for important information on which Business Attributes can be updated and which cannot. See the *Universal Routing Business Process User's Guide* for examples of Business Attribute usage.

Using Business Attributes

In IRD you can use Business Attributes in the following objects:

- The Generic Segmentation object (see [page 383](#)), where you can select Business Attributes when constructing an expression (see [Figure 49](#)):

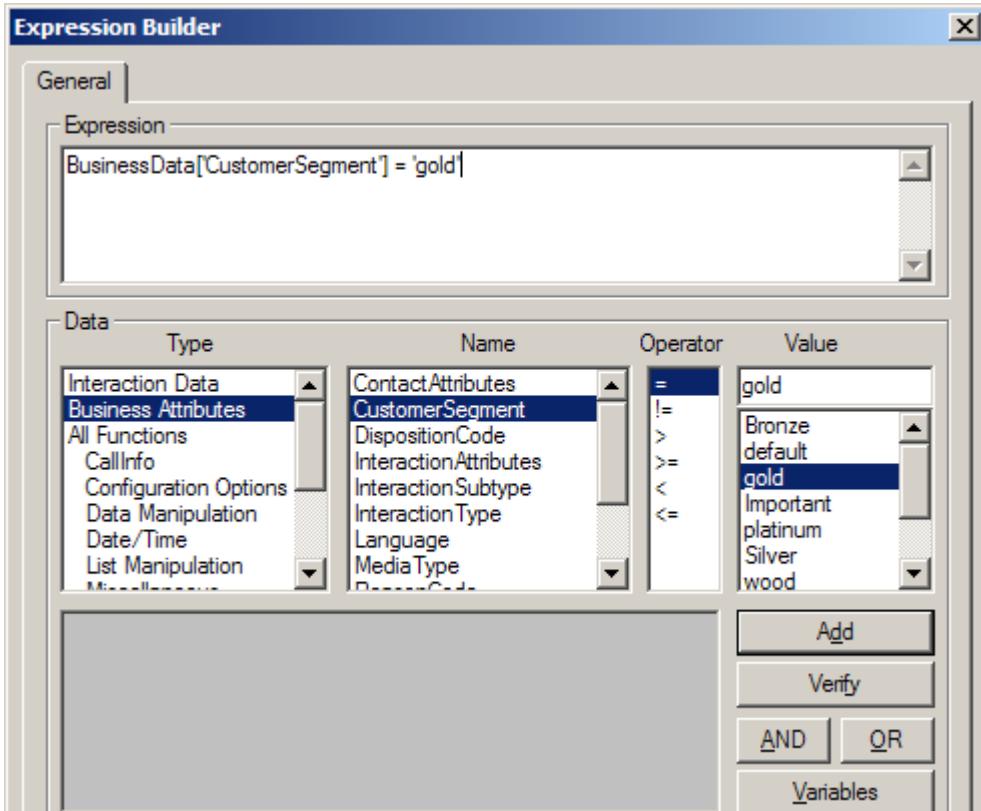


Figure 49: Generic Segmentation Object Using Business Attributes

- The If (see [page 129](#)) and Assign objects (see [page 120](#)), where you can select Business Attributes when constructing an If statement.
- Certain Function object (see [page 127](#)) functions use Business Attributes, including the Attach (see [page 451](#)), Update (see [page 467](#)), and UpdateBusinessData (see [page 468](#)) functions.
- The MultiAttach Miscellaneous object ([page 140](#)), where you can attach Business Attributes to interactions (see Figure 78 on [page 141](#)).

Note: For information on how to use Business Attributes in Multimedia objects, see the *Universal Routing Business Process User's Guide*. Also see the multimedia routing strategy samples in *Universal Routing 7.6 Strategy Samples*.

Using Objects in Strategies

For ease of reference, the sections ahead describe the various strategy building objects in alphabetical order by object category:

Strategy-Building Objects Organization

The remainder of this chapter provides detailed descriptions of the strategy-building objects listed in [Table 7](#).

Table 7: Strategy-Building Objects

Data and Services Objects	Miscellaneous Objects	Multi-media Objects	Out-bound Objects	Routing Objects	Segmentation Objects	SMS Objects	Workflow and Res. Management Objects	Voice Treatment Objects
Database Wizard (page 108)	Assign (page 120)	Acknowledgement (page 161)	Add Record (295)	Default (page 316)	ANI (page 374)	Create SMS Out (page 386)	Force Logout (page 393)	Busy (page 407)
External Service (page 115)	Call Subroutine (page 121)	Attach Categories (page 178)	Do Not Call (299)	Force Routing (page 316)	Business (page 375)	Send SMS Out (page 390)	Set Agent DND State (page 395)	Cancel Call (page 407)
Web Service (page 119)	Error Segmentation (page 123)	Autoresponse (see page 181)	Processed (302)	Load Balancing (page 317)	Classify (page 377)		Set Agent Media State (page 398)	Collect Digits (page 407)
	Entry (page 126)	Chat Transcript (page 191)	Reschedule (page 305)	Percentage (page 318)	Date (page 380)		Set Multimedia Strategy State (page 400)	Delete User Announcement (page 409)
	Exit (page 127)	Classify (page 198)	Update Record (page 307)	Queue Interaction (page 320)	Day of Week (see page 380)			Fast Busy (page 409)

Table 7: Strategy-Building Objects (Continued)

Data and Services Objects	Miscellaneous Objects	Multi-media Objects	Out-bound Objects	Routing Objects	Segmentation Objects	SMS Objects	Work-flow and Res. Management Objects	Voice Treatment Objects
	Function (page 127)	Create Email Out (page 212)		Route Interaction (page 322)	DNIS (page 381)			IVR (page 409)
	If (page 129)	Create Interaction (page 205)		Selection (page 326)	Generic (page 383)			Music (page 410)
	Multi-Assign (page 140)	Create Notification (page 217)		Service Level (page 333)	Screen (page 384)			Pause (page 411)
	Multi-Attach (page 140)	Create SMS (page 221)		Statistics (page 346)	Time (page 385)			Play Announcement (page 411)
	Macro (page 130)	Find Interaction (page 228)		Switch-to-Strategy (page 347)				Play Announcement and Collect Digits (page 412)
	Multi-Function (page 146)	Forward E-Mail (page 233)		Workbin (page 348)				Play Application (page 412)
		Identify Contact (page 237)		Workforce (page 353)				RAN (play recorded announce.) (page 413)
		MultiScreen (page 241)						Record User Announcement (page 414)
		Redirect E-Mail (page 250)						Ringback (page 415)

Table 7: Strategy-Building Objects (Continued)

Data and Services Objects	Miscellaneous Objects	Multi-media Objects	Out-bound Objects	Routing Objects	Segmentation Objects	SMS Objects	Work-flow and Res. Management Objects	Voice Treatment Objects
		Render Message Content (page 254)						Set Default Destination (page 415)
		Reply E-Mail from External Resource (page 258)						Silence (page 415)
		Screen (page 263)						Text-to-Speech (page 415)
		Send E-Mail (page 269)						Text-to-Speech and Collect Digits (page 416)
		Stop Interaction (page 120)						Verify Digits (page 416)
		Update Contact (page 281)						
		Update Interaction (page 285)						
		Update UCS Record (page 290)						

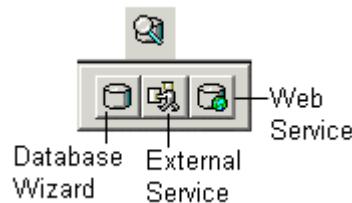
Table 7: Strategy-Building Objects (Continued)

Data and Services Objects	Miscellaneous Objects	Multi-media Objects	Out-bound Objects	Routing Objects	Segmentation Objects	SMS Objects	Workflow and Res. Management Objects	Voice Treatment Objects
		Submit New Interaction (page 277)						
		Distribute Custom Event (page 226)						

Note: In voice strategies, you usually configure a Segmentation object first. Non-voice routing strategies are different. See *Universal Routing 7.6 Strategy Samples* for more information.

Data and Services

Clicking the Data and Services icon in the Routing Design window opens a subtoolbar containing three buttons (see [Figure 50](#)).

**Figure 50: Data and Services Buttons**

- The Database Wizard button places an object that you can use to retrieve data from both SQL databases and (through a Custom Server) non-SQL databases. For more information, see “Database Wizard” on [page 108](#).
- The External Service button enables you to exchange data with third party (non-Genesys) servers. For more information, see “External Service” on [page 115](#).
- The Web Service button enables you to send requests to a Web Service outside of Genesys and use the results you receive in a routing strategy. For more information, see “Web Service” on [page 119](#).



Database Wizard

When you place the Database Wizard object in a strategy and double-click it, the Database Wizard launches.

- You can configure the Database Wizard to retrieve information from a database using a SQL SELECT statement or by invoking a SQL-stored procedure for standard SQL databases.
- The Database Wizard can also get information from non-SQL database using a configured Custom Server.

The first wizard dialog box is where you specify whether to use a Database Server or Custom Server (see [Figure 51](#)).

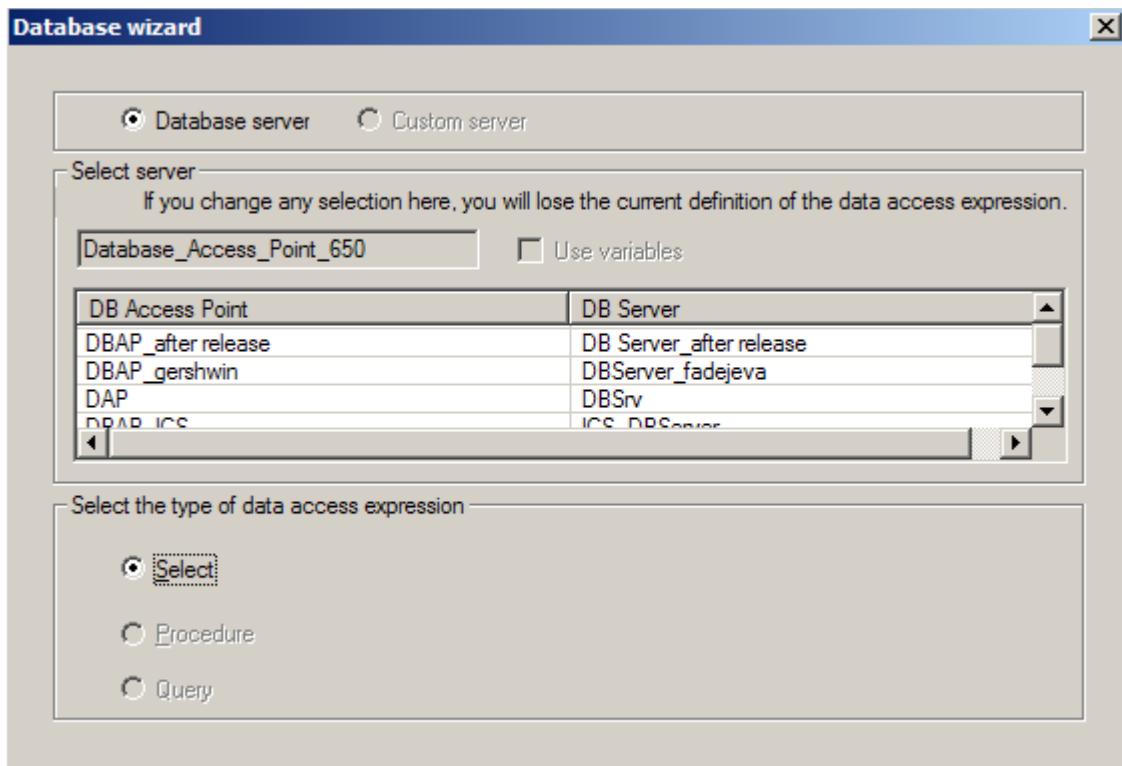


Figure 51: Database Wizard, Type of Database Access

The first wizard dialog box (see [Figure 51](#)) is also where you select a DB Access Point/DB Server (see “[default_db_server](#)” on [page 637](#) for more information) and then specify either a SELECT statement, Procedure, or Query (restricted to a SELECT statement as described on [page 113](#)).

Assume you clicked Select followed by the Next button.

Note: Although not shown in the figures ahead, the bottom of the next four wizard dialog boxes contain the following buttons: Back, Next, Cancel, Help. The final wizard dialog box adds a Finish button.



The second wizard dialog box is where you select the database table and fields (see [Figure 52](#)).

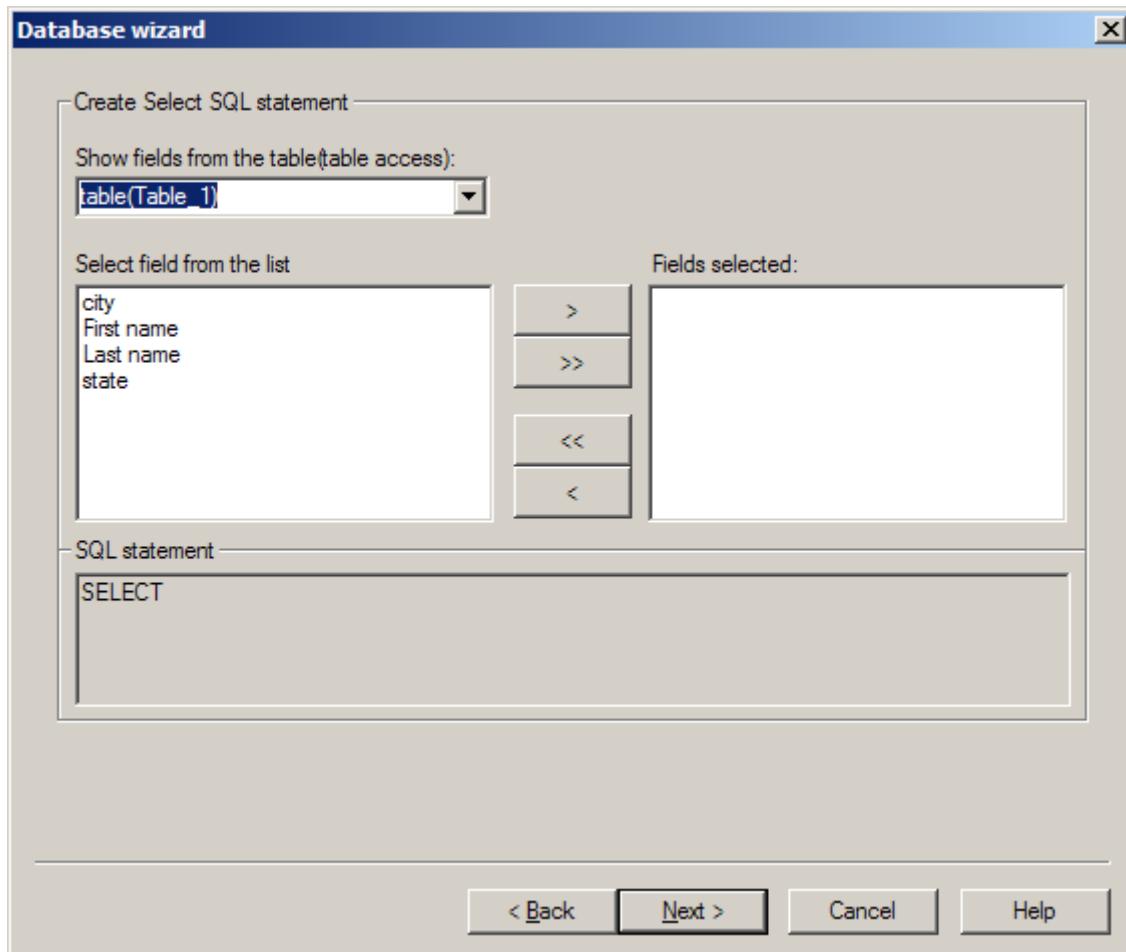


Figure 52: Database Wizard, Constructing Select Statement

The next wizard dialog box is where you construct the SELECT statement (see [Figure 53](#)).

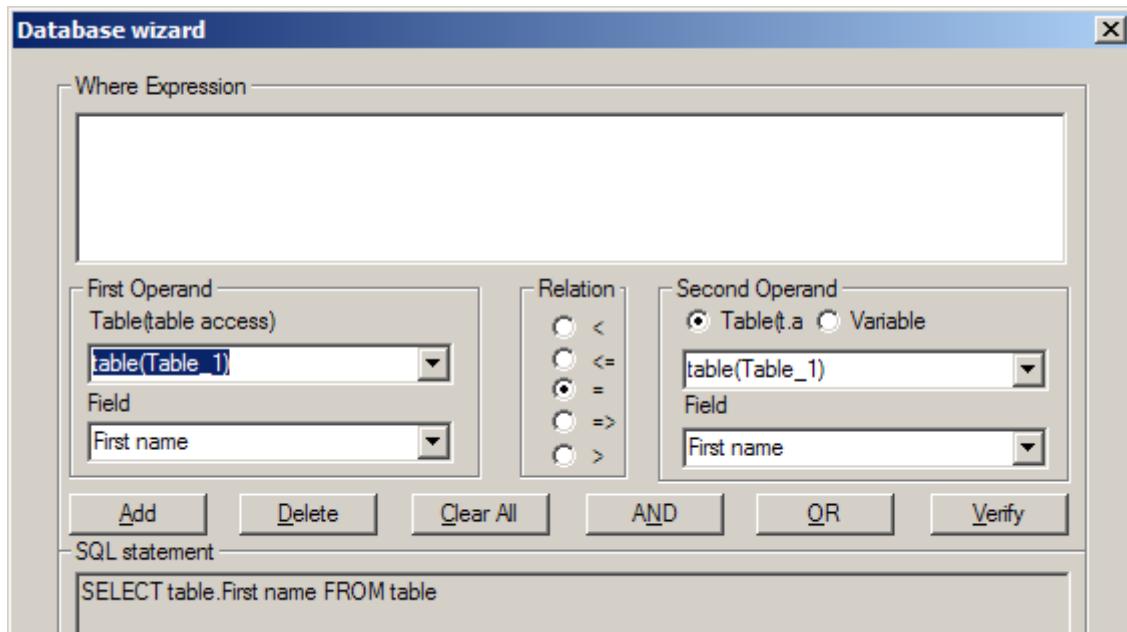


Figure 53: Database Wizard, Constructing Where Clause

The next wizard dialog box is where you specify sort criteria (see [Figure 54](#)).

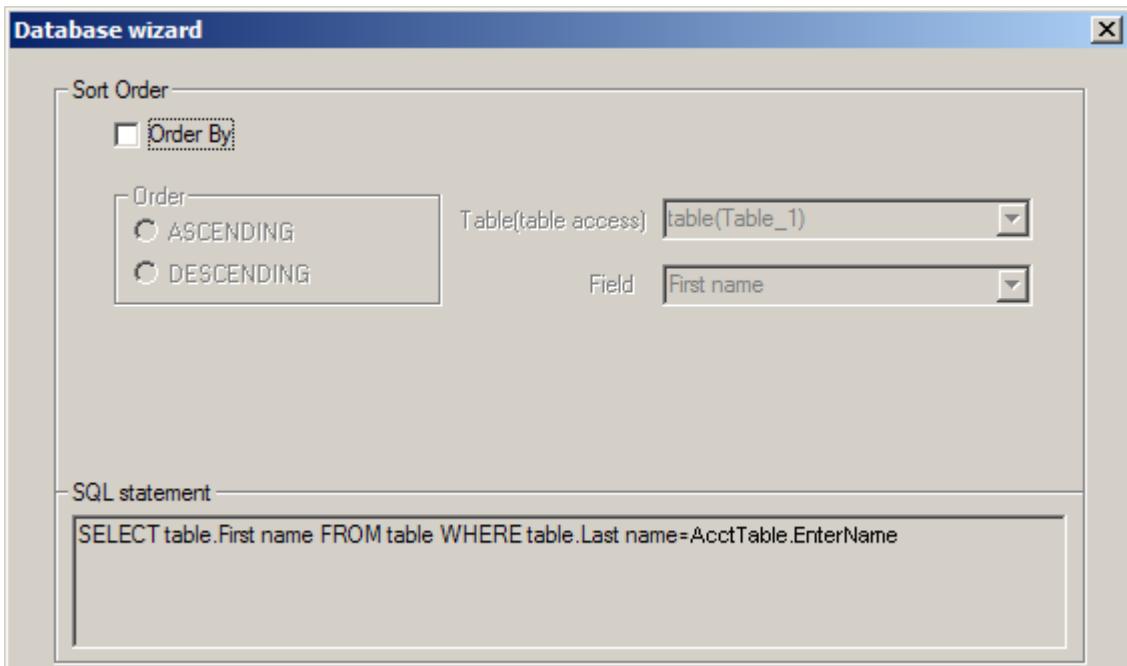


Figure 54: Database Wizard, Sort Order

Since the data read from the database must be retained for subsequent use in the strategy, the next wizard dialog box lets you assign data to variables or to attach data to the call (see [Figure 55](#)).

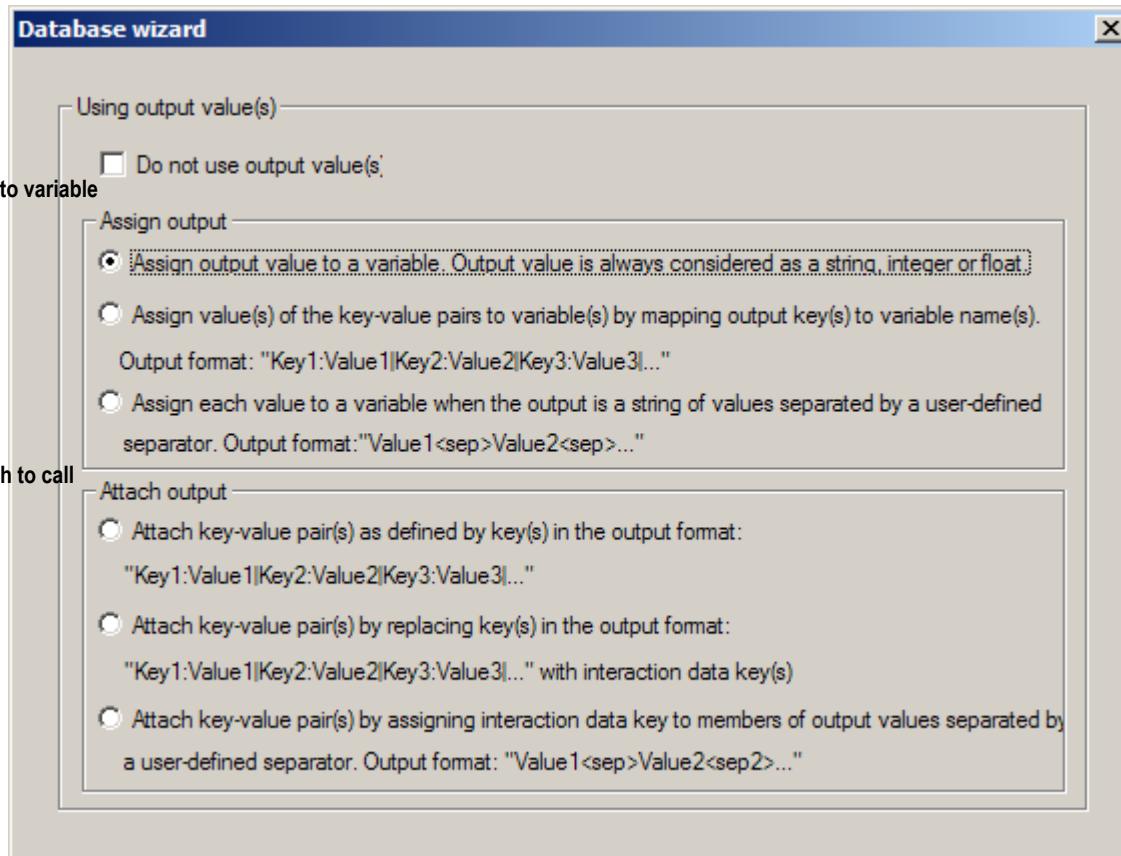


Figure 55: Database Wizard, Assigning or Attaching Output

If you assigned the output to a variable, the next wizard dialog box is where you can edit the variable. (see [Figure 56](#)).

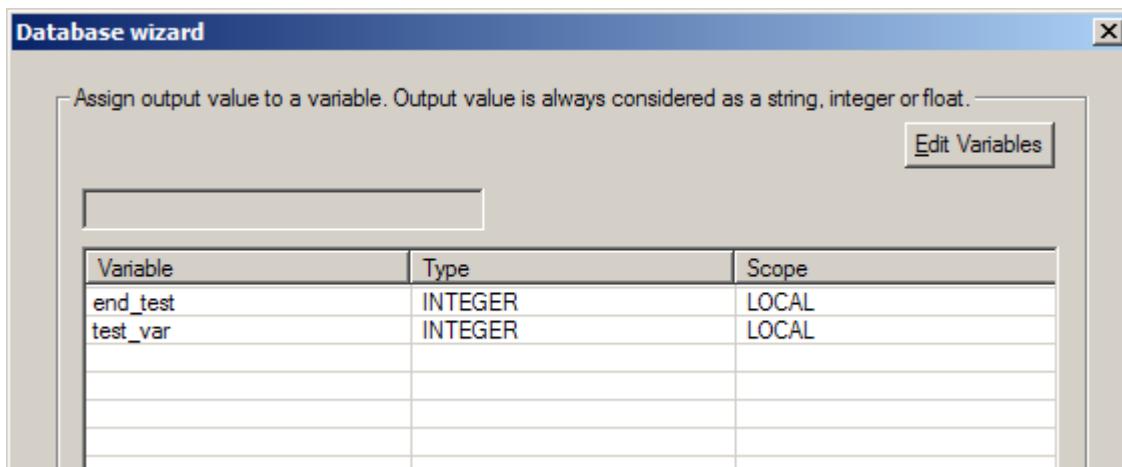


Figure 56: Database Wizard, Assigning Output Value to a Variable

When through in the Database Wizard, click the Finish button.

Warning! When using the Database Wizard to query a database, the maximum size of a SQL statement created by Database Wizard cannot exceed 1010 bytes.

Possible Error Messages

The following error messages are possible:

- `0001 Unknown Error`—for example, a failure to send the request to DB Server.
- `0013 Remote Error`—error/unknown message from DB Server.
- `0014 Unknown server`—the specified server is not found or the interaction has no permission to use it.
- `0015 Closed server`.

Database Access Point

If you choose the Database server option shown in Figure 51 on [page 108](#), you must select the Database Access Point (DAP). DAP is a Configuration Layer object that defines a DB Server together with a database and the connection parameters for the database. See [page 637](#) for information about using `_DEFAULT_`. For details about the configuration of a DAP, see the *Framework 8.1 Configuration Manager Help*.

SELECT Statements

The `SELECT` statement (see Figure 53 on [page 110](#)) created in the Database Wizard contains the name of one or more tables in a relational database and the name of one or more columns in that table. It must also contain one or more `WHERE` clauses. (This is the only case of a `SELECT` statement that can be directly generated by the Database Wizard.) The `SELECT` statement will retrieve the fields from every record in the table that satisfies the `WHERE` clause condition. The `SELECT` statement cannot write to a database.

Warning! Check your `SELECT` statement carefully after using the Create Select SQL Statement dialog box (see Figure 52 on [page 109](#)). The wizard can sometimes truncate a Table Access name in the `FROM` clause.

Stored Procedures

The Procedure option (see Figure 51 on [page 108](#)) in the Database Wizard creates a precompiled sequence of SQL statements that retrieve data from

and/or write data to a database. It may require one or more arguments that define or limit the data to be retrieved or written.

Query

The Query option (see Figure 51 on page 108) provides read access to the selected database by allowing you to execute queries before including them in SELECT statements or stored procedures. Figure 57 shows the dialog box where you type in the SELECT statement.

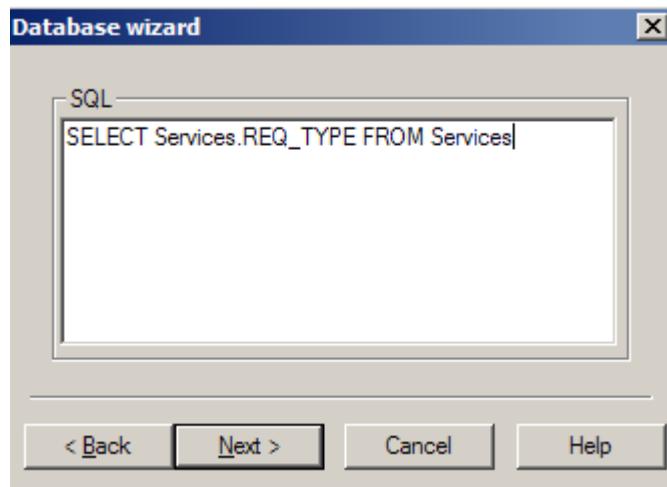


Figure 57: Database Wizard Dialog Box for Creating Query

Warning! Use only a SELECT statement in a query. Do not use any other commands, such as CREATE TABLE or INSERT. Even though you are able to type in these commands, Genesys does not support such commands (as well as other commands requiring a write to database operation). String constants used in strategies cannot contain quotes. Do not place a space after an equal sign. You cannot use multiple-line SQL statements in Query mode; you must enter SQL statements as single lines.

Tables and Fields

When configuring tables and fields in Configuration Manager that are used in the Database Wizard, be certain to match the names of the tables and fields in the database exactly. These names are case-sensitive.

Note: The table names and field names in databases are limited to letters, numbers, and underscores and must begin with a letter. Names cannot contain spaces or special characters.

Custom Server

Custom Server (see Figure 51 on [page 108](#)) allows URS to retrieve and send data from a non-SQL database using a custom procedure. The IRD interface for defining a custom procedure is identical to the interface used for a SQL-stored procedure.

See the *Universal Routing 8.1 Deployment Guide* for information on configuring and installing Custom Server and for information on Custom Server options.

Error Handling

If DB Server or Custom Server shuts down while URS is waiting for a response from one of them, all interactions are routed to the red port of the Database object. If a backup server is configured, interactions will still be routed to the red port until the backup server is functioning.

Use the Error Segmentation object (see Figure 64 on [page 124](#)) connected to the red port to route interactions in case DB Server or Custom Server shuts down.

Customizing URS for Last Agent Routing

You can customize Universal Routing to supply the identity of the agent who last interacted with a customer, enabling you to route the customer's next interaction to that agent.

To accomplish this, you need the following:

- A desktop application with the ability to update the customer contact database with the identity of the last agent to handle the customer.
- A strategy that includes a lookup in the Universal Contact Server database for the last agent information for the customer and uses that agent as the first routing target.

Note: Routing an interaction to the last agent who handled a specific customer also requires specifying a reasonable timeout for this agent target. If the timeout expires, you can use a multi-queueing mechanism to queue the interaction to multiple targets.



External Service

Use in non-voice routing strategies to exchange data with third party (non-Genesys) servers that use the Genesys Interaction SDK or any other server/application that complies with Interaction Server communication protocol (see “URS Communication with Interaction Server” on [page 151](#)).

Note: In order to use this object, the third party server/application must already be defined in the Configuration Manager Application folder as a server of type Third Party Server or Third Party App. Before completing the object properties dialog box, you must already know the names of Services, Methods, and Signatures (requested input/output parameters) provided by the external service.

External Service General Tab

[Figure 58](#) shows the General tab of the External Service Properties dialog box.

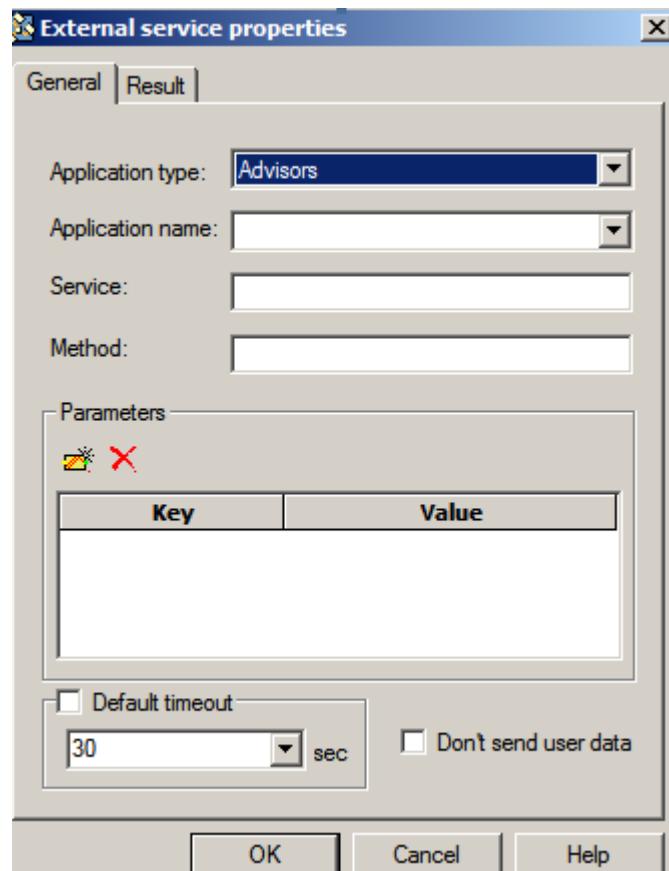


Figure 58: External Service Object, General Tab

Use the information in [Table 8](#) to complete the fields in the General tab.

Table 8: External Service Object, General Tab

Parameter	Description
Application Type	Click the down arrow and select either Third Party App or Third Party Server. In order to select from the complete list of Applications available in your Configuration environment, you must add the following option to Default section of your IRD Application in Configuration Manager: Option name: tools_tuneup Option value: extended
Application Name	Enter the name of the third party application or server. This must be the same name entered for the server in the Applications folder of Configuration Manager. IRD provides integrity checking on the name.
Service	Required. Enter the name of the Service defined by the third party server or application for the functionality requested.
Method	Enter the Method defined by the third party server or application.
Parameters	Click the button to add a new entry:  0 A row appears under the Key and Value columns. <ul style="list-style-type: none"> • Click under the Key column to display a down arrow. Click the down arrow and enter an input parameter name. • Click under the Value column and enter a value or select the name of the variable that contains the value. If you change your mind, click the X to delete the row.
Default Timeout	Check to use the default of 30 seconds. If not checked, URS uses the Reconnect Timeout entered for third party server or application in the Configuration Manager properties dialog box, Server Info tab. In the case of a connection or service request failure, error codes are returned. (This checkbox must be unchecked to see the sec drop-down menu.)
Don't sent user data	Check to enable sending user data when using this object. The user data is sent only if the checkbox is unchecked (default).

External Service Result Tab

[Figure 59](#) shows the Result tab of the External Service Properties dialog box.

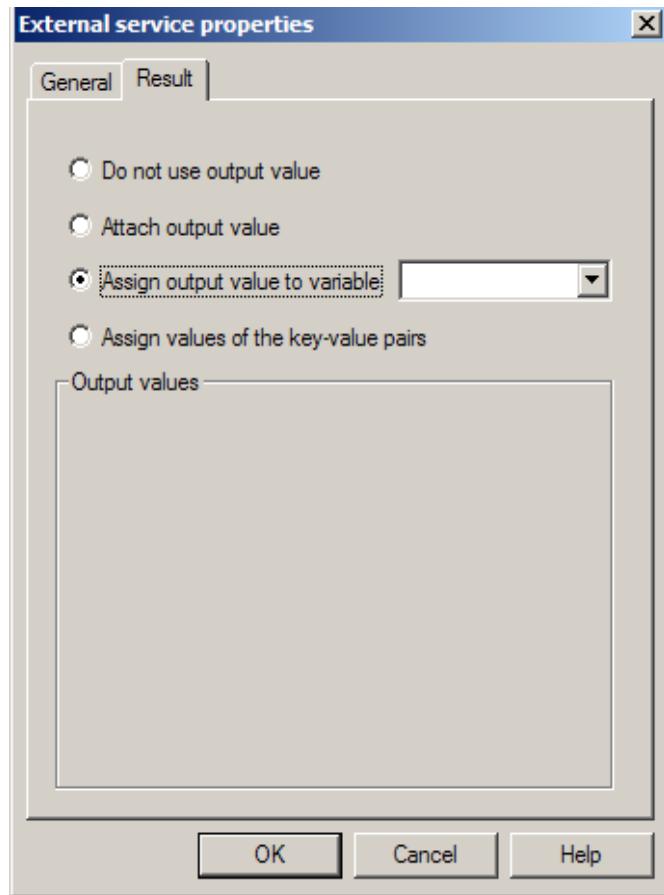


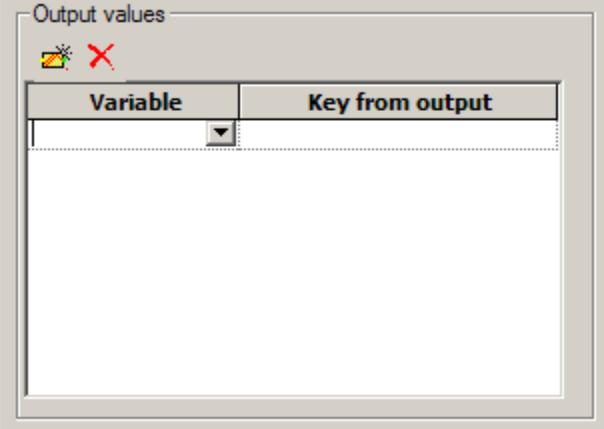
Figure 59: External Service Object, Result Tab

Use the information in [Table 9](#) to complete the Result tab (see [Figure 59](#)) in the External Service Properties dialog box.

Table 9: External Service, Result Tab

Parameter	Description
Do not use output value	Click to not use the returned values in any of the ways listed below. URS will ignore the parameters data received from the server.
Attach output value	Default. Click to attach returned values to the interaction.
Assign output value to a variable	Click to assign any returned output values to a variable, which must already be defined (see “Variables in Objects and Expressions” on page 92). URS will create a string composed of the parameters data and assign this string to the selected variable from the dropdown menu.

Table 9: External Service, Result Tab (Continued)

Parameter	Description
Assign values of the key-value pair	Click to assign values in the returned array of output parameters to a list of variables (see “Returned Results for the Acknowledgement Object” on page 173). If you select this option, the Result tab displays the Output Values area.
Output values	 <p>Click the button to add a new entry.</p>  <p>A row appears under the Variable and Key from output columns.</p> <ul style="list-style-type: none"> Click under the Variable column to display a down arrow. Click the down arrow and select a variable for storing the output. Click under the Key from output column and enter a key to identify the output parameter. <p>If you change your mind, click the X to delete the row.</p>

Error and Fault Codes

The External Service object transparently sets whatever error code the External Service or Interaction Server returns. There are also some predefined error codes that Interaction Server returns if there is a problem executing External Service (see [Table 10](#)).

Table 10: Predefined Interaction Server Error Codes

Error Code	Description
1	Application with specified type “type” not found
2	Application with specified name “name” not found

Table 10: Predefined Interaction Server Error Codes (Continued)

Error Code	Description
3	Application with specified type “type” and name “name” not found
4	Third party server response timeout
5	Can’t establish connection to specified application

The External Service object uses the standard third party server fault codes shown in [Table 11](#).

Table 11: External Service Object Fault Codes

Fault Code	FaultString Value
6	Service name is not specified.
7	Method name is not specified.
8	Service name -“name” is incorrect.
9	Method name – “name” is incorrect.

Note: In the case of an error from T-Server or Interaction Server, use the `GetLastErrorInfo[]` function to access the error information (see [page 617](#)).



Web Service

Use the Web Service object to interact with Web-based applications (Web Services) outside of Genesys applications. You may wish to do this, for example, if you are using the *Gplus* Adapter for mySAP Data Access Component. For information on using this object, see Appendix B, “IRD Web Service Object” on [page 771](#).

Miscellaneous Objects

Figure 60 shows the buttons for Miscellaneous objects that appear when you click the Miscellaneous icon.

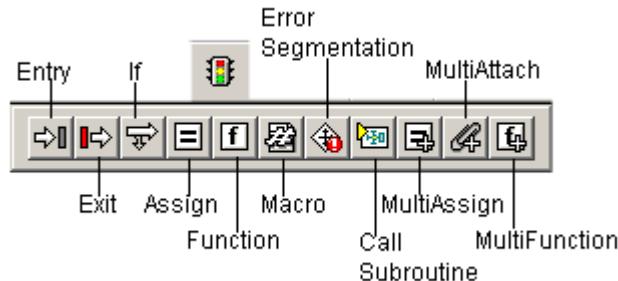


Figure 60: Miscellaneous Objects Toolbar



Assign

The Assign object gives a value to a variable that has been predefined in the Variable List dialog box (see Figure 41 on [page 93](#)). The variable can then be used as a parameter of almost any object. Figure 61 shows an expression in Assign Properties dialog box when assigning the returned value of the UData function to a variable.

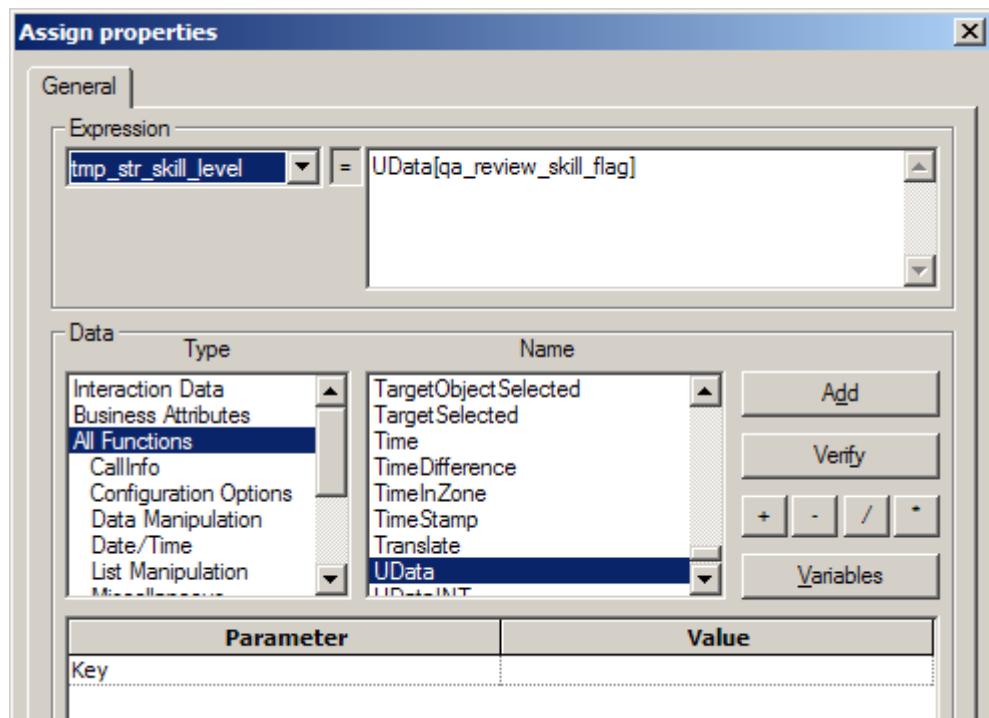


Figure 61: Assign Properties Dialog Box

For mathematical symbols allowed in expressions, see [page 91](#).



Call Subroutine

Note: For more about subroutines, see [page 65](#).

Using the Call Subroutine object you can call a subroutine from within a strategy or another subroutine. When you place the Call Subroutine object in a strategy and open its dialog box, you can select a subroutine. [Figure 62](#) shows the dialog box after selecting a subroutine.

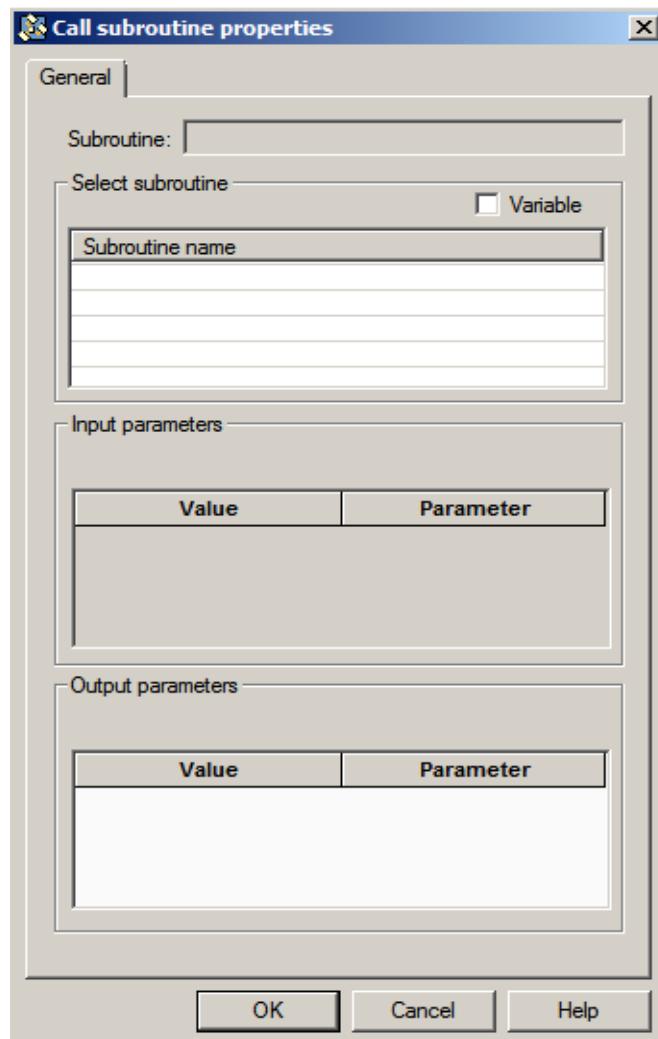


Figure 62: Call Subroutine Dialog Box

When this object is encountered in a strategy, control of the interaction is passed to the subroutine (see [page 65](#)) called by the object.

Values can be passed between a strategy and a subroutine through input and output parameters. Call Subroutine objects can accept 0, 1, or more input

parameters and return 0, 1, or more output parameters. The number of input parameters do not have to equal the number of output parameters. The number depends on what values you want passed into and out of the object from the subroutine. The number of output parameters for the Call Subroutine object **must** equal the number of output parameters for the subroutine.

These parameters can be constants, interaction data values, or variables. Their types can be `string`, `float`, `integer`, `Date`, `Time`, or `Day`.

Note: If a constant name is entered as a parameter that is the same name as a variable, the constant will be interpreted as a variable.

Call Subroutine Using a Variable

In the Call Subroutine object properties, you can switch to a list of variables and select one that has the name of subroutine to be called as its value (in run time). An extendable list box for input and output parameters appears when the `Variable` check box is enabled (in the `Select subroutine` section of the Call Subroutine properties). The variable must have the name of the subroutine that has the same number of parameters as those configured in the Call Subroutine object.

Example of a Call Subroutine Scenario

A strategy assigns the variable `Balance` to a customer's account balance and the variable `RecentCharges` to the charges recently made. When the subroutine was created, two input parameters and two output parameters were also created. In the Call Subroutine object, the `Balance` variable is assigned to one input parameter and the `RecentCharges` variable is assigned to the other.

When a Call Subroutine object is encountered in a strategy, the values for these variables are passed to the subroutine through input parameters. During the subroutine, the values of these variables are added and assigned to a new variable, `NewBalance`. The passed `NewBalance` value may be used to route the interaction. If the interaction is not routed, the `NewBalance` value is passed through an output parameter back to the original strategy.

Important Information on the Call Subroutine Object

- Variables are specific to the subroutine and cannot be used by other subroutines or strategies, including the strategy that calls the subroutine.
- When using the Call Subroutine object, be careful that you do not design the subroutine so that it calls the original strategy in which the Call Subroutine object is located, thus causing an infinite loop.
- If the subroutine input/output parameters are changed, you must correct and recompile the strategy.

If an interaction is not routed to the default destination in the subroutine, the control of the interaction returns to the original strategy. URS continues applying actions to the interaction based on the objects that follow the Call Subroutine object.

If an interaction is routed in the subroutine, any object that is capable of generating a **0008** error code (routing done) when placed after the subroutine (postrouting) will generate the error code **0008**.

The following error messages are possible:

- **0018**—Unknown object. If a subroutine call has failed (**018** error), then output parameters are undefined.
- **0020**—Error in subroutine—this error code is reported to the calling strategy after an interaction reaches an Exit object if an error occurs within a subroutine. When this error occurs, the interaction is routed using the red output port on the Call Subroutine object. This error does not explain where the error occurs in the subroutine, only that an error occurred



Error Segmentation

The error-handling mechanism is embedded within objects whenever applicable. The Error Segmentation object offers an optional connection for further error processing. This object is used by connecting the error (red) output port from another non-segmentation object to the input port of Error Segmentation. Connecting an object to Error Segmentation overrides standard behavior for a particular error. This enables further processing of errors and prevents the interaction from going to the default destination.

The Error Segmentation object processes error codes generated by functions and objects. [Figure 63](#) shows the properties dialog box dropdown menu where you can select one of the functions described in Chapter 3, “Interaction Routing Designer Functions,” on [page 423](#).

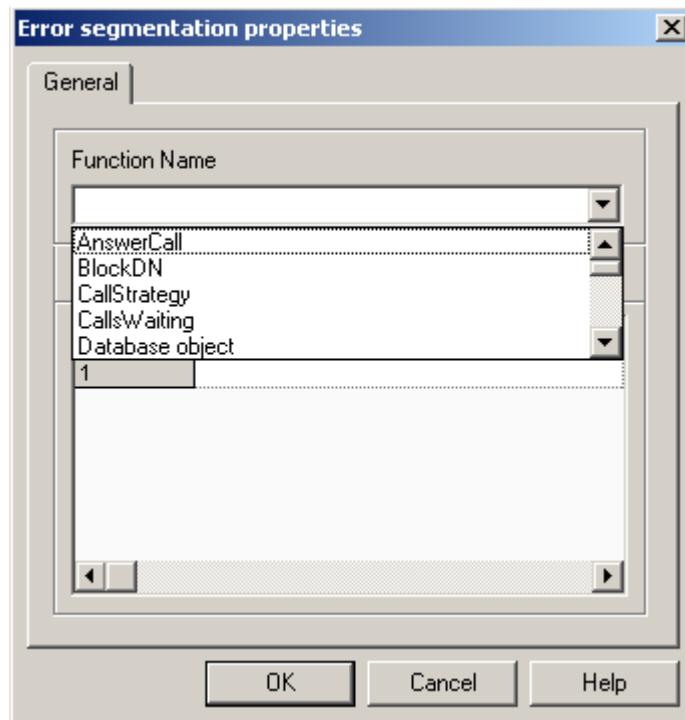


Figure 63: Error Segmentation Object, Selecting a Function

Figure 64 shows the properties dialog box after selecting the DeliverCall function (see [page 591](#)) and error 0003 Treatment in progress (see [page 591](#)).

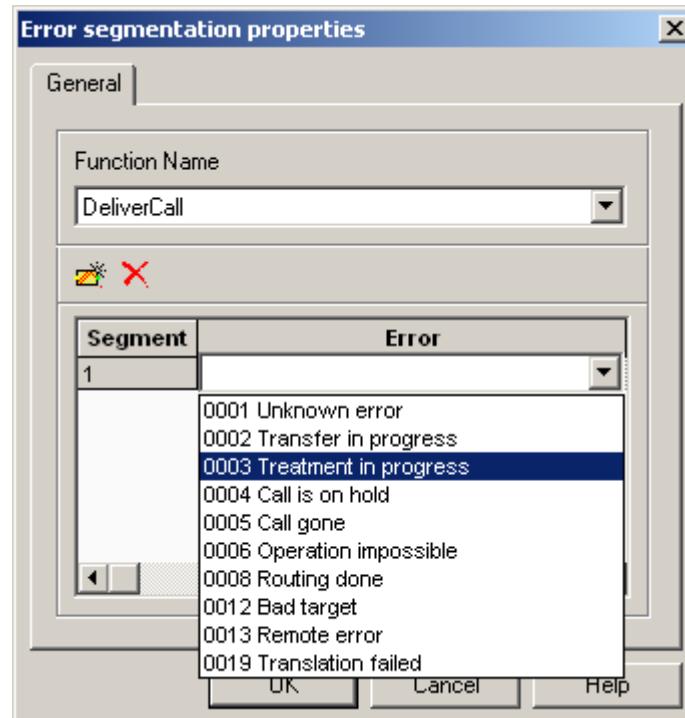


Figure 64: Error Segmentation Properties Dialog Box

The possible error codes are listed in the description of each function. For more information, see [Table 12](#).

Error Codes

Error codes are made up of an error number and error message. The first four characters of an error code form a digital code. They are followed by a space and an error message, for example, 0013 Remote Error.

[Table 12](#) provides the list of error codes used by IRD and URS.

Table 12: Error Codes

Error Number	Error Message
-001	Timeout
-002	Negative answer
0000	No error
0001	Unknown error
0002	Transfer in progress
0003	Treatment in progress
0004	Call is on hold
0005	Call is gone
0006	Operation impossible
0008	Routing done
0012	Bad target
0013	Remote error
0014	Unknown server
0015	Closed server
0017	Invalid request
0018	Unknown object
0019	Translation failed
0020	Error in subroutine

Table 12: Error Codes (Continued)

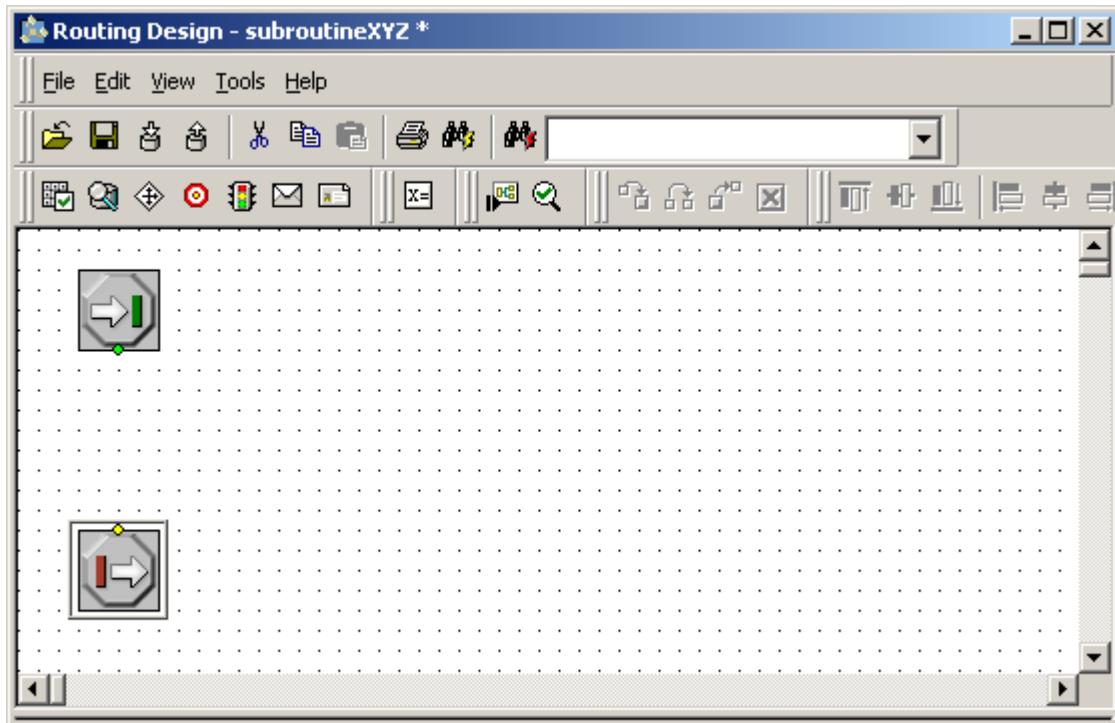
Error Number	Error Message
0021	Out of resources
0022	No data

If error 0013, `Remote Error`, is returned, you can use the `GetLastErrorInfo` object to obtain more information about the error. See “`GetLastErrorInfo`” on page 518 for more information.

Note: “`SetLastError`” on page 546 is one of the functions that use the above error codes.

**Entry**

When you create a new strategy or subroutine, IRD automatically places an Entry object in the upper-left corner of the strategy (see Figure 65).

**Figure 65: Entry and Exit Objects**

The Entry object indicates the starting point of the strategy.



Exit

When you create a new subroutine, IRD automatically places an Exit object in the subroutine in the lower-left corner of the subroutine (see [Figure 65](#)). The Exit object is used to terminate the strategy or to return from the subroutine, if any, to the strategy.

Note: You must use the Exit object in the green port branch of `RequestRedirectCall`. If function `SendRequest` (`RequestRedirectCall`), the `RedirectCall` macro, or the `RedirectCallMakeNotReady` macro is used within a strategy, you must end the green port branch of those objects with the Exit object.



Function

When configuring a Function object, you select one of the functions described in Chapter 4, “Interaction Routing Designer Functions” on [page 423](#).

If applicable, use the Expression box in the Function object to create an expression, which assigns the function output to a variable.

When you place the Function object in your strategy and double-click it, the `Function Properties` dialog box opens. [Figure 66](#) shows an example completed dialog box.

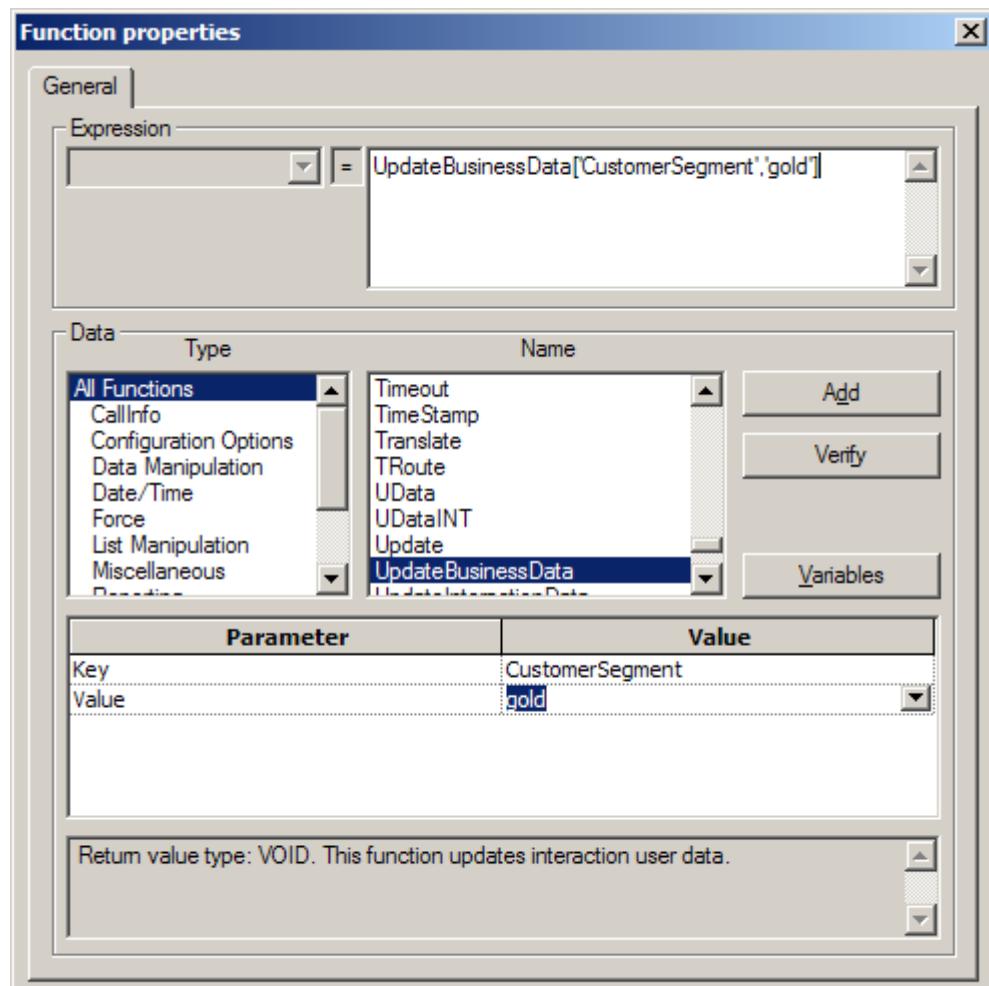


Figure 66: Function Properties Dialog Box

Use the Function Properties dialog box to select a function and enter values for the parameters of that function. Functions can also be accessed within an expression.

General Rules for Using the Function Object

Here are some general rules for using the Function object:

- To specify function parameters, use the Parameter-Value grid of the Function object (see [Figure 66](#)). The grid provides a list of possible parameter values from configuration data (where applicable) and/or a list of variables defined in the strategy. No matter how the parameter value is specified, IRD first considers it as a variable name. If the strategy doesn't have variable with such a name, then the parameter value is considered a constant and IRD automatically appends quotes.

- You can also use the multi-line edit box to specify/modify function parameters and the function itself. Although this way is more flexible, no automatic help from IRD is provided. In this case, you are responsible for correct syntax; for example: enclosing constants in quotes, etc.
- Use the **Verify** button to check the syntax of an entered expression. IRD will also perform syntax checking when you click the **OK** button and provide a warning message in the case of any syntax errors with the option to fix the error immediately or fix it later.

For an alphabetical summary of available functions, see Table 130, “Interaction Routing Designer Functions,” on [page 430](#). After the alphabetical summary, the function descriptions in Chapter 3, “Interaction Routing Designer Functions,” on [page 423](#) are grouped alphabetically by function Type (see Figure 66 on [page 128](#)).



If

Use the If object properties dialog box to create an If expression. [Figure 67](#) shows the dialog box after constructing an If statement.

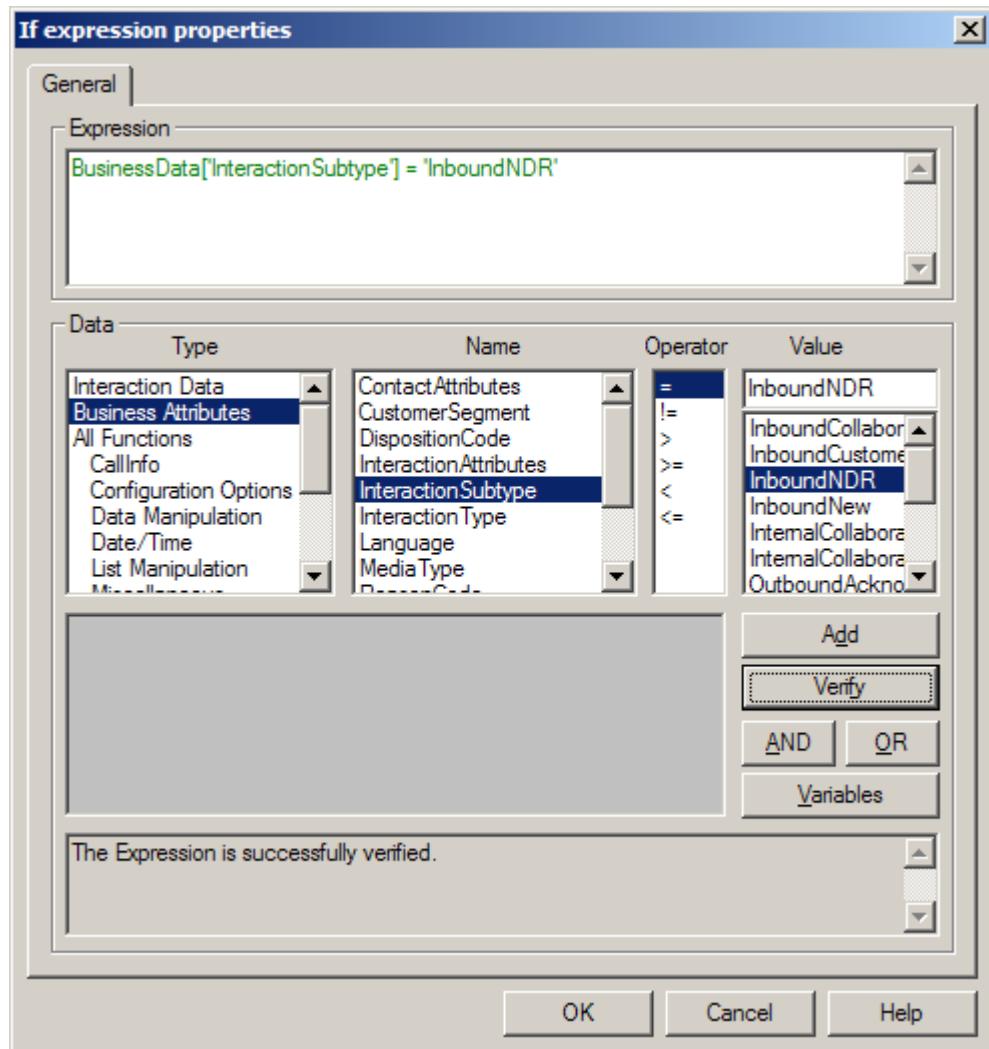


Figure 67: If Expression Properties Dialog Box

You can create expressions using Interaction Data (see [page 72](#)), Business Attributes (see [page 98](#)), and functions (see [page 423](#)).

See “GetMediaType” on [page 459](#) for information on returning the media type assigned by T-Server. See “GetMediaTypeName” on [page 474](#) for information on returning the media type name in the Configuration Server database.



A macro is an object that allows you to combine several actions, functions, and expressions (optional) into one reusable block. It is both a re-usable object and a strategy-building object. You create a macro from one or more IRD objects including their parameters.

Note: If you modify the body of a macro you must recompile and resave all strategies that use this macro in order to make those changes take effect in the strategies.

A macro works like a user-defined function. The same object (or set of objects) and/or parameters can be defined once in a macro and can then be repeatedly re-used. Once a macro is defined, you simply place the macro object in a strategy, and select the macro from the list of available macros. You do not have to repeatedly enter the same set of objects and/or parameters. IRD supplies the following predefined macros:

1. CheckBusinessRule
2. ComplexSample
3. DelimitTargetList
4. MakeAgentNotReady
5. RedirectCall (uses function “SendRequest” on [page 534](#))
6. RedirectCallMakeNotReady (uses function “SendRequest” on [page 534](#))
7. SimpleSample

[Figure 68](#) shows the predefined macros:

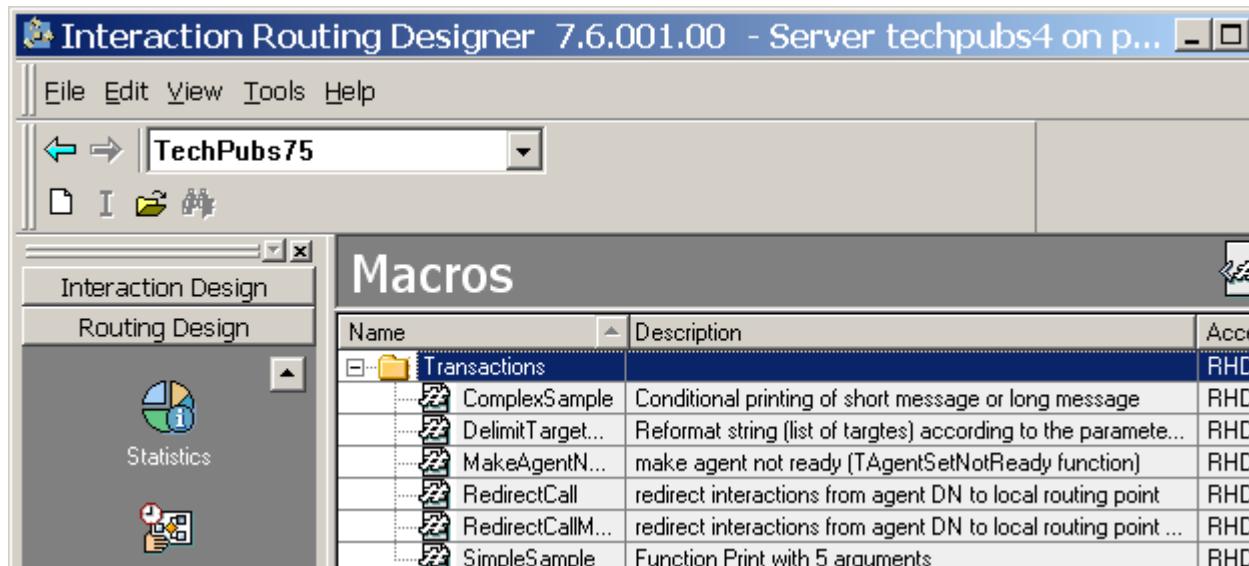


Figure 68: IRD Predefined Macros

If a macro row appears green, this indicates it is used in a strategy.

Understanding the Predefined Macros

In addition to using the SendRequest function (see [page 534](#)), three of the seven predefined macros also use T-Library functions defined in *Voice Platform SDK 8.0 .NET API Reference* or *Voice Platform SDK 8.0 Java API*.

Reference. You may also wish to consult the *Genesys Events and Models Reference Manual*. The IRD macros that use T-Library functions are listed in [Table 13](#):

Table 13: IRD Macros and T-Library Functions

IRD Macro Name	T-Library Function
MakeAgentNotReady	TAgentSetNotReady
RedirectCall	TRedirectCall
RedirectCallMakeNotReady	TRedirectCall, TAgentSetNotReady

Open IRD and follow the steps below to review the MakeAgentNotReady predefined macro.

1. In Macro view, double-click MakeAgentNotReady. You will see that Description references the T-Library TAgentSetNotReady function (see [Figure 69](#)).

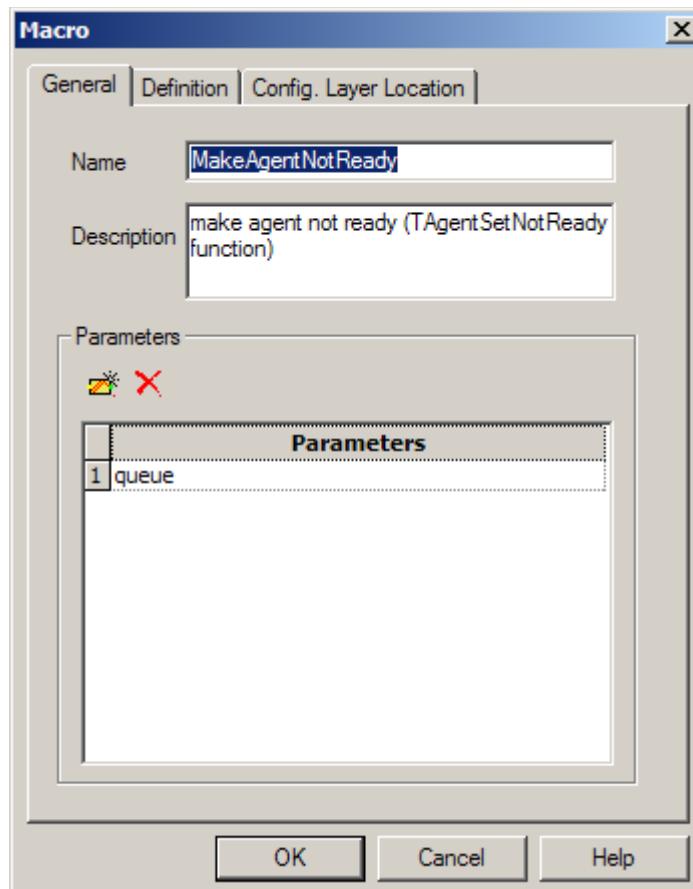


Figure 69: MakeAgentNotReady Macro, General Tab

1. Under Parameters, note the queue parameter. You enter this parameter when you use the macro in a routing strategy.
2. Click the Definition tab.
3. Note that the macro also uses the IRD SendRequest function (see Figure 70).

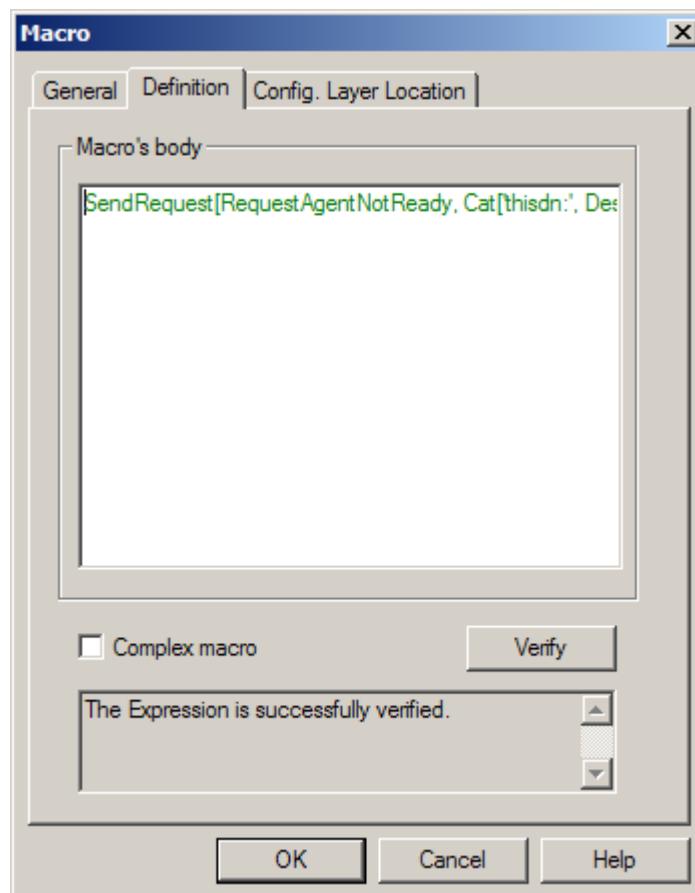


Figure 70: MakeAgentNotReady Macro, Definition Tab

The function “SendRequest” on [page 534](#), available for selection in the IRD Function object (see [page 127](#)), allows URS to send T-Library Request messages to T-Server. SendRequest takes two parameters: Type and Request.

[Figure 71](#) shows the IRD Function object with the SendRequest function selected. The Type dropdown menu lists available T-Library Request message types.

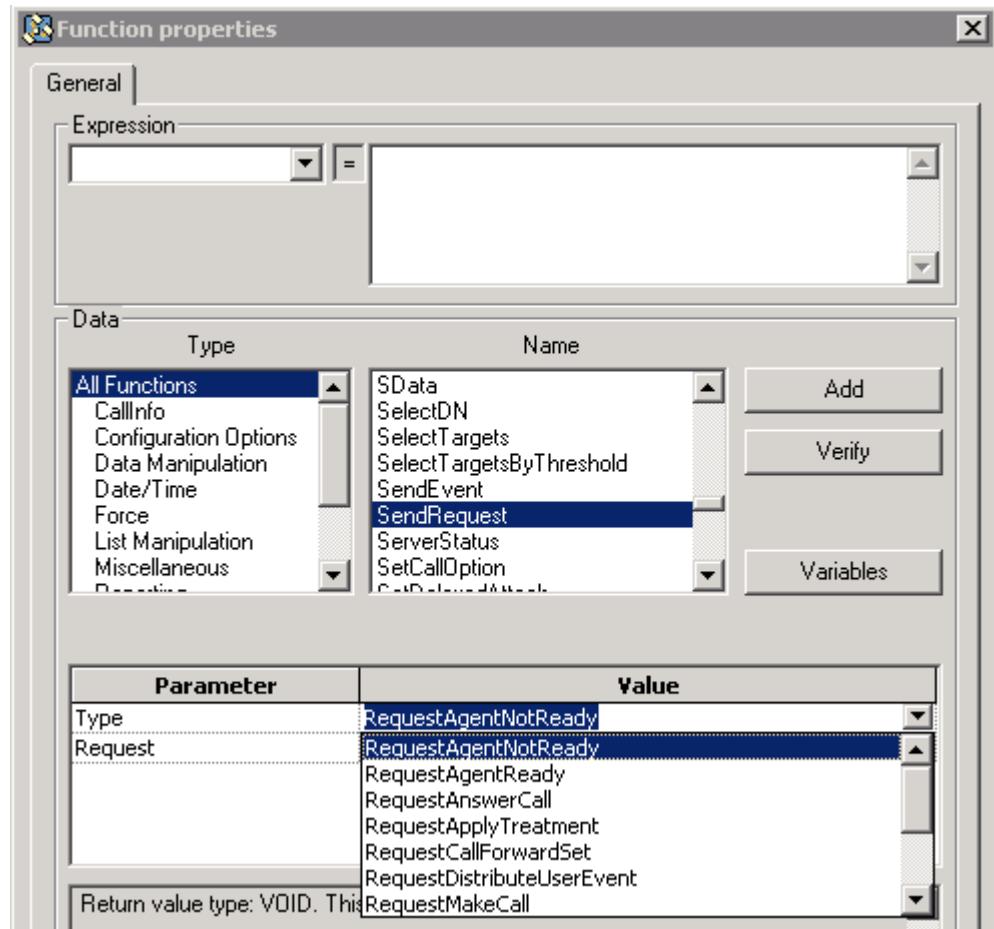


Figure 71: Function Object Properties Dialog Box For SendRequest

RequestAgentNotReady in Figure 71 uses the T-Library TAgentSetNotReady function documented in the *Voice Platform SDK 8.0 .NET API Reference* or *Voice Platform SDK 8.0 Java API Reference*.

Similarly, the next function in the dropdown, RequestAgentReady uses the T-Library TAgentSetReady function, and so on.

The full expression in the body of the macro is

SendRequest[RequestAgentNotReady, Cat['thisdn:', Dest[], '|21:', queue, '|16:', 6]]. The number 21 maps to AttributeThisQueue, which you can find by looking up TAttribute and locating 21 in the enumerated list of attributes. The number 16 maps to AttributeAgentWorkMode, also in the enumerated list. The number 6 can be found by looking up TAgentWorkMode, which shows that AgentWalkAway=6.

4. Note the Macro's body area in the Macro Definition tab (see Figure 70 on page 133). To get more detail on the parameters specified after SendRequest under Macro's body, refer to *Voice Platform SDK 8.0 .NET API Reference* or *Voice Platform SDK 8.0 Java API Reference*, TAgentSetNotReady function.

5. Under Macro's body, note the queue parameter. It maps to the queue parameter defined in the General tab (see Figure 69 on [page 132](#)) as a parameter that the macro user will input. You are prompted for a queue when you place the Macro object in a strategy, open its properties dialog box, and select the MakeAgentNotReady macro.

Using a Macro

Use MakeAgentNotReady as follows:

1. In the Routing Design window (see Figure 3 on [page 29](#)), under Miscellaneous objects, click the button for the Macro object (see Figure 60 on [page 120](#)).
2. Place the Macro object in your strategy.
3. Double-click to open its properties dialog box.
4. Select MakeAgentNotReady.
5. Enter a value for the queue parameter.
6. Click Add, and then Verify. The Expansion area shows the macro and its parameters (see [Figure 72](#)).

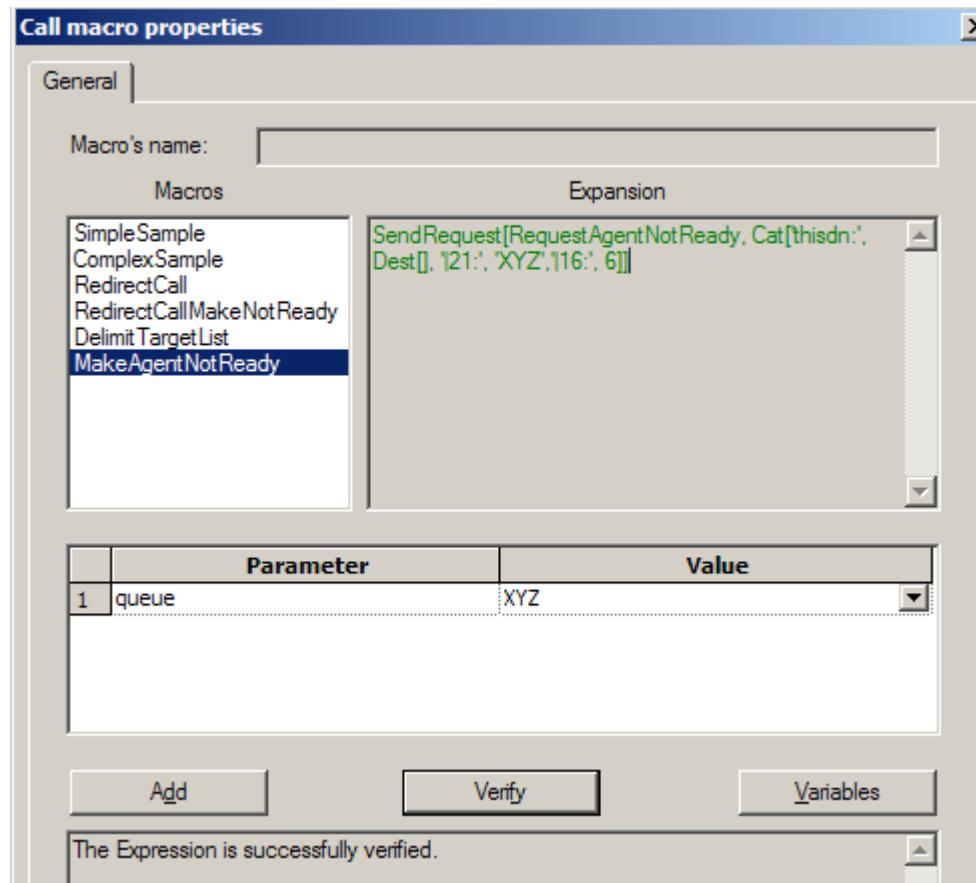


Figure 72: Call Macro Properties Dialog Box

Simple Versus Complex Macros

The Definition tab of the Macro Properties dialog box (see Figure 70 on [page 133](#)) has a Complex Macro check box. IRD supplies one predefined complex macro, ComplexSample (see [Figure 73](#)).

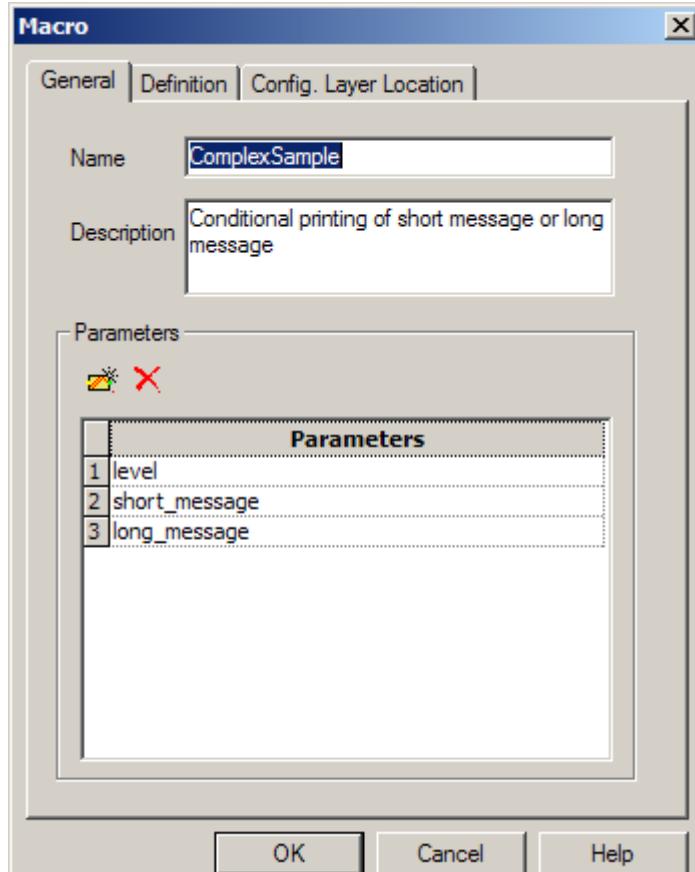


Figure 73: ComplexSample Predefined Macro, General Tab

With the exception of ComplexSample, all the other predefined macros (see Table 13 on [page 132](#)) are simple macros. The difference between the two is as follows:

- A simple macro is a sequence of actions, one after the other, usually (but not necessarily) performed by IRD objects.
- A complex macro is an arbitrary multiline fragment of strategy written in plain text format. The use of conditions is optional. If needed, they can be defined in the Expression Builder (see Figure 39 on [page 88](#)), to determine whether to perform an action. If the condition is true, an action is performed, such as executing a function call.

When entering the Macro's body area for a complex macro, one option is to cut and paste from the Expression Builder (see Figure 39 on [page 88](#)).

- Separate the condition and the action with a hyphen (-) and a forward angle bracket (>).

- Use a pound (#) sign for comments.

Figure 74 shows the Macro's body area for the predefined macro, ComplexSample.

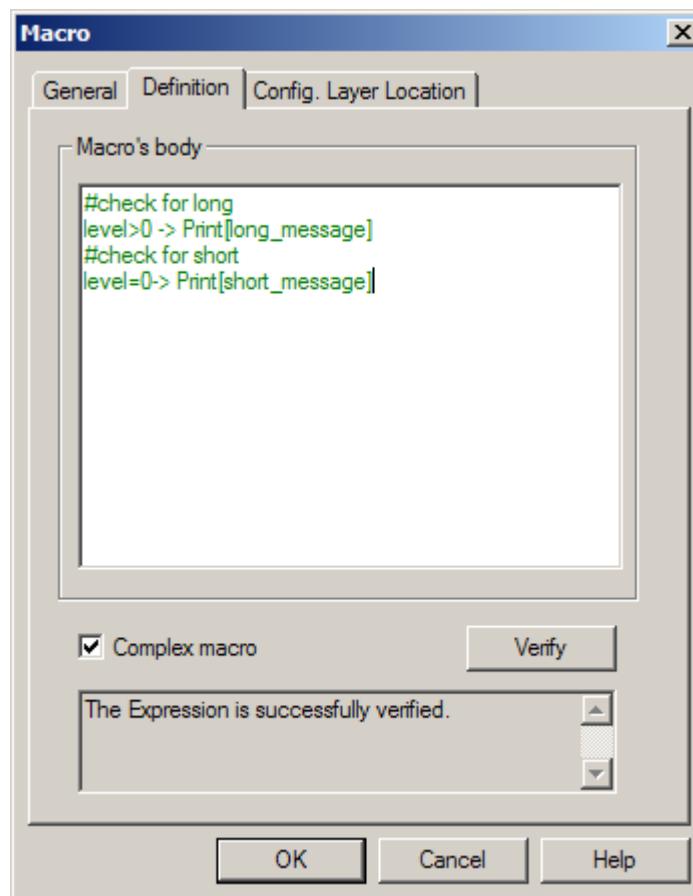


Figure 74: ComplexSample Predefined Macro, Definition Tab

Notes:

- On the Definition tab, use the Verify button to check the validity of the created macros. IRD uses two compilers which make the verifications. Depending on the syntax used in the macro definition, IRD calls the corresponding compiler (icd_compiler.exe or es_compiler.exe) to compile the macros and report errors or provide warnings.
 - For step-by-step instructions on creating macros, see *Universal Routing 8.1 Interaction Routing Designer Help*.
-

Using Macros to Handle Ring-No-Answer

You can use the `RedirectCall` and `RedirectCallMakeNotReady` macros to handle ring-no-answer situations if your switch hardware and particular Genesys T-Server supports redirecting calls on ring-no-answer. See the function “SendRequest” on [page 534](#) for more information as well as information on custom macros.

CheckBusinessRule Macro

The `CheckBusinessRule` macro accept the business rule name as an input parameter and is executed in the following ways:

- If a business rule is evaluated and found to be true, it is executed through the green port.
- If it is evaluated and found to be false, it is executed through the red port.
- If there are no existing business rules, or if the existing business rules are in the wrong format (for example, not saved with IRD 8.1 or later), it is executed through the red port.

DelimitTargetList Macro

Note the `DelimitTargetList` macro in Figure 68 on [page 131](#). Figure 75 shows the General and Definition tabs for this macro.

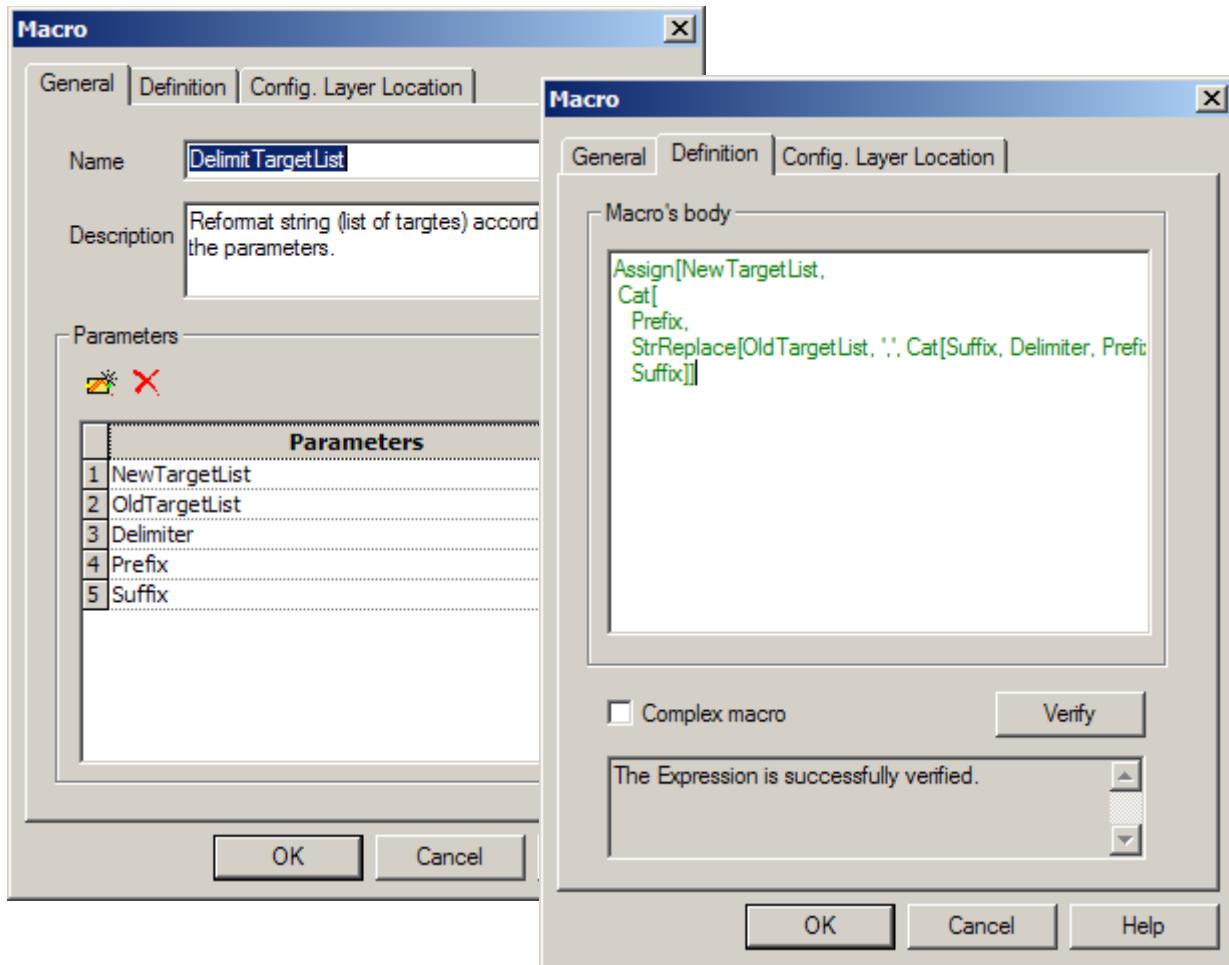


Figure 75: DelimitTargetList Macro

The `DelimitTargetList` macro allows you to reformat an output list of targets based on replacement parameters that you enter. It provides the ability to specify:

- A new target list (variable to keep the results of target list reformatting).
- The old target list (string or variable for comma-separated target list such as one returned by “`SelectTargetsByThreshold`” on [page 530](#)).
- The new `Delimiter` to be used as separators for each item on the reformatted target list.
- Start (`Prefix`) and end (`Suffix`) characters to pre-pend and append to each element.

You can use the `DelimitTargetList` macro to check/reformat the output returned from a database (without the need to manipulate the output data using String functions), or simply to create a properly formatted input target list for another operation.



MultiAssign

The MultiAssign object works like the Assign object. The difference is that with MultiAssign you can assign/edit multiple variables that have been predefined in the Variable List dialog box (see Figure 41 on [page 93](#)). The variables can then be used as a parameter of almost any object. Figure 76 shows the dialog box after assigning a value to a variable.

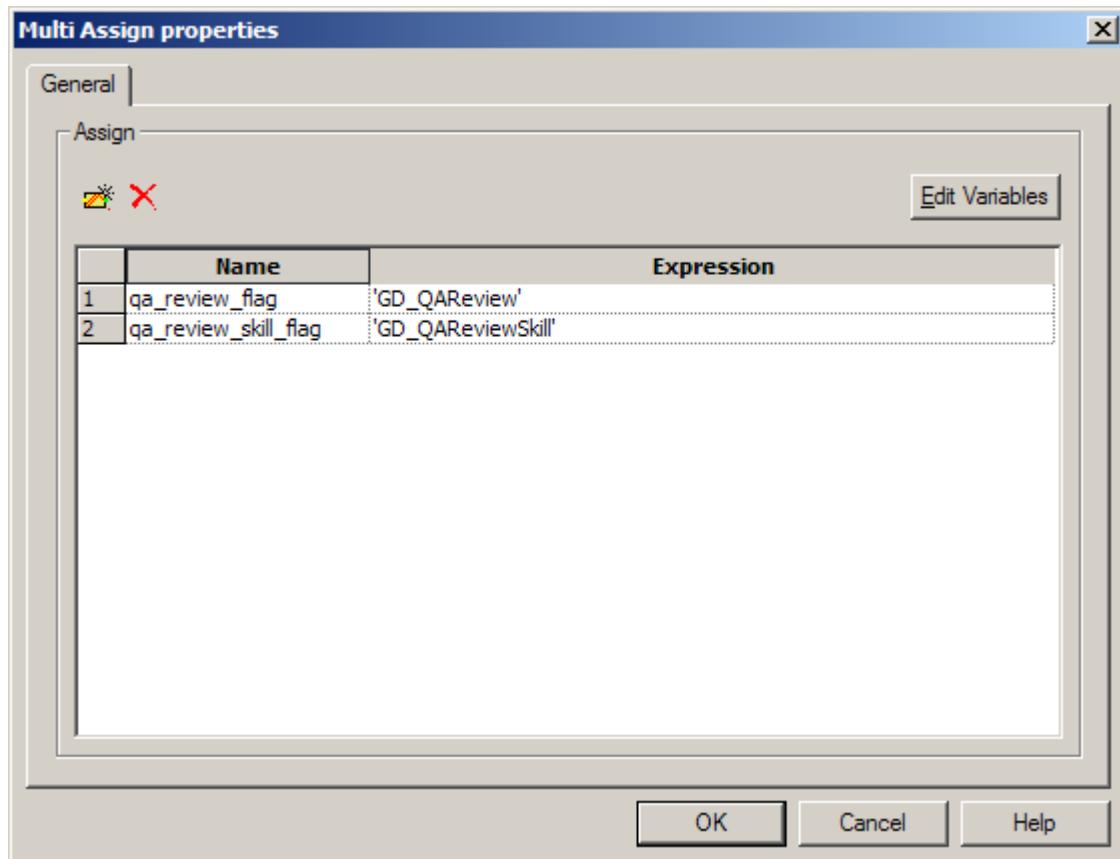


Figure 76: MultiAssign Properties Dialog Box



MultiAttach

Note: Also see the Update function and “Attaching Several Key-Value Pairs” on [page 467](#).

Use this object to request (RequestAttachUserData) T-Server to:

- Attach or update one or more pieces of data to the interaction, such as account or menu selection data that the customer entered via an IVR.
- Attach the Requested Skill (see Figure 77 on [page 141](#) and “Requested Skills” on [page 142](#)).

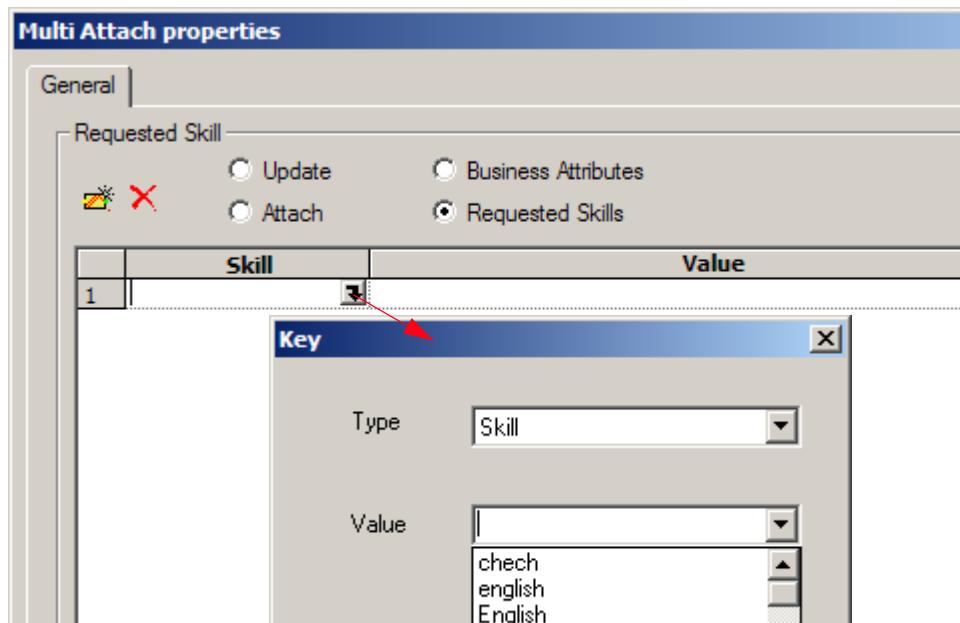


Figure 77: MultiAttach Object, Requested Skills Option

- Attach Business Attributes (see [page 98](#)) to the interaction, such as Service Type, Customer Segment, and Service Objective (see [Figure 78](#)).

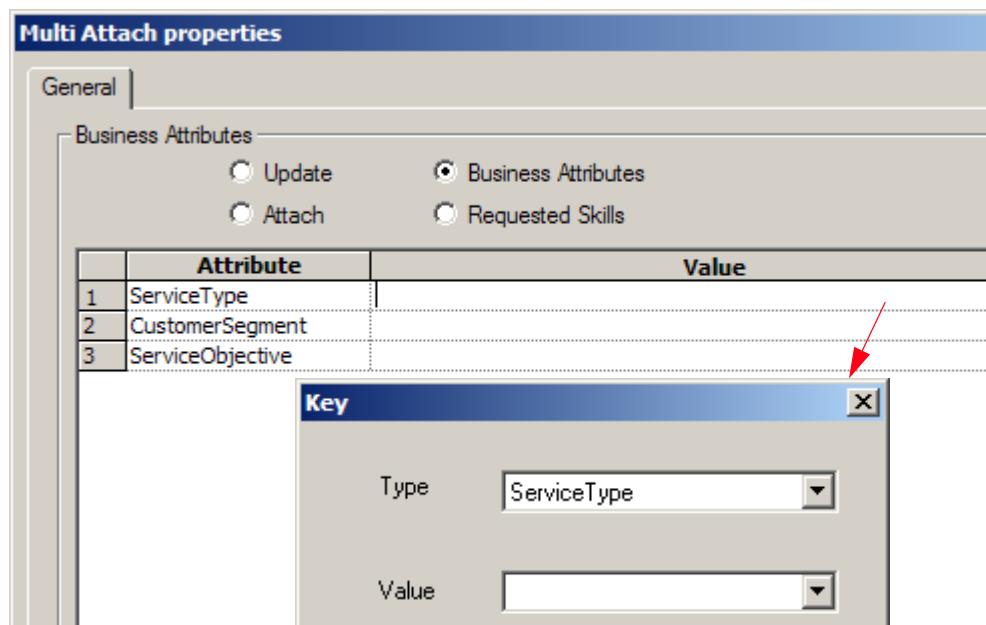


Figure 78: MultiAttach Properties Dialog Box

Also see “Business Attributes Option” on [page 143](#).

Warning! You cannot attach Business Attributes in a 7.0 Framework environment, which uses Enumerators as attributes.

Requested Skills

The MultiAttach object Requested Skills option (see Figure 77 on [page 141](#)) works like this:

- When the strategy enters the MultiAttach object, URS attaches the RRequestedSkills key-value list with the specified skill/value pairs.
- When the call is finally routed, URS checks if there is an RRequestedSkills list present in the call data. If it is there, URS copies the RRequestedSkills content into the RRequestedSkillCombination key-value pair (converting the RRequestedSkills list into a single string value), and attaches it to the call.

The idea behind this functionality is to first allow attaching the ideal skill set to the call. Then later, if the ideal skill set could not be found, once the call is finally routed, the interaction contains both (1) information about the original desired skill set and (2) the actual skill set used when the call was routed.

You can specify multiple skills and skill levels (see Figure 79).

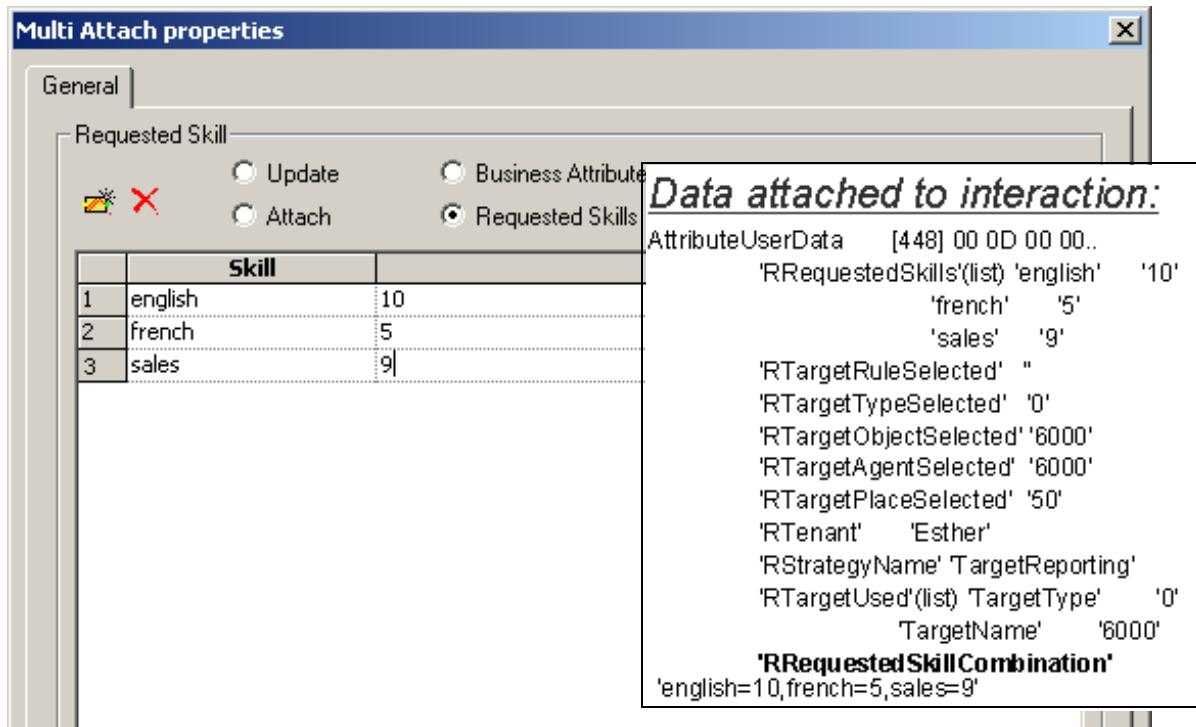


Figure 79: Requested Skills Attached to Interaction

Business Attributes Option

The Business Attributes option in the MultiAttach Properties dialog box (see Figure 78 on [page 141](#)) lets you assign Business Attributes to incoming interactions. These attributes can serve as the baseline for a routing decision. They can also be used to generate reports that focus on business objectives. The Business Attributes that can be attached with this object are:

- Customer Segment (for example, Gold and Silver customers)
- Service Type (for example, Sales or Service requests)
- Service Objective (time objective to service a voice interaction).

The next section discusses the above Business Attributes. For information on attaching other Business Attributes, see the functions “Attach” on [page 451](#) and “Update” on [page 467](#).

Customer Segment

Customer Segment categorizes a customer based on their revenue potential to the enterprise relative to a business line. For example, in the credit card business line, customers are categorized according to their maximum spending limit. This is indicated by whether the customer has a Platinum (high), Gold (medium), or Bronze (low) credit card.

A customer can belong to one or multiple Customer Segments depending on the value against which they are being measured.

For example, in the banking industry with multiple business lines, a customer can belong to the most highly valued Customer Segment (platinum) when their credit worthiness is being measured.

However, the same customer can belong to a lower Customer Segment (bronze) if measured as a total asset to the bank. This can happen if the customer does not conduct any other business with the bank, such as securing a mortgage or obtaining a loan.

From reporting perspective, the Customer Segment value of an interaction can change over time depending on how many times the interaction is transferred between business departments.

Service Type

Service Type describes what service a customer is requesting at a particular moment in time. For example, an IVR system may have the customer select “1” for Loan, “2” for Investment or “3” for Information. Loan, Investment, and Information are all Service Types. The exact value for each Service Type is defined in Configuration Manager.

A contact center can use Service Type to categorize interactions according to their product or service offerings. If it is possible for an interaction to request multiple Service Types, a contact center can combine both to represent the interaction. Examples:

TechSupport_DSLResidential, TechSupport_DSLCommercial.

Service Objective

In general, Service Objective is the time objective to service a voice interaction. During deployment, and as an ongoing maintenance activity based on customer needs, the contact center defines a Service Objective for each combination of Customer Segment, Service Type, and Media Type.

- For voice interactions, the Service Objective is called “*answering objective*”. It is the target time for the voice call to be answered by a live agent. “Answered” refers to the necessary action taken by the agent to start talking to the customer. Answering objectives for voice interactions are defined in terms of seconds and minutes.
- For e-mail interactions, Service Objective is called *response objective*, which is the target time for responding to the interaction. The term *respond* refers to the necessary action taken by the agent to send an e-mail reply or complete the interaction. Response objectives for asynchronous interactions are defined in terms of minutes, hours, or days.

Assigning Business Attributes

When the **Business Attributes** radio button (see Figure 78 on [page 141](#)) is selected, **Attribute** and **Value** columns appear. Click the down arrow to bring up the **Key** dialog box. Here you can keep the attribute in the **Type** field or select **Variable**. Values for Customer Segment, Service Type, and Media Type are predefined in Configuration Manager under **Business Attributes**. See “The Interaction Design Window” on [page 98](#) for more information.

Using Configuration Manager **Objective Tables** (under **Tenant**), you have the option of defining baseline Service Objectives for various combinations of Media Type, Service Type, and Customer Segment (see [Figure 80](#)).

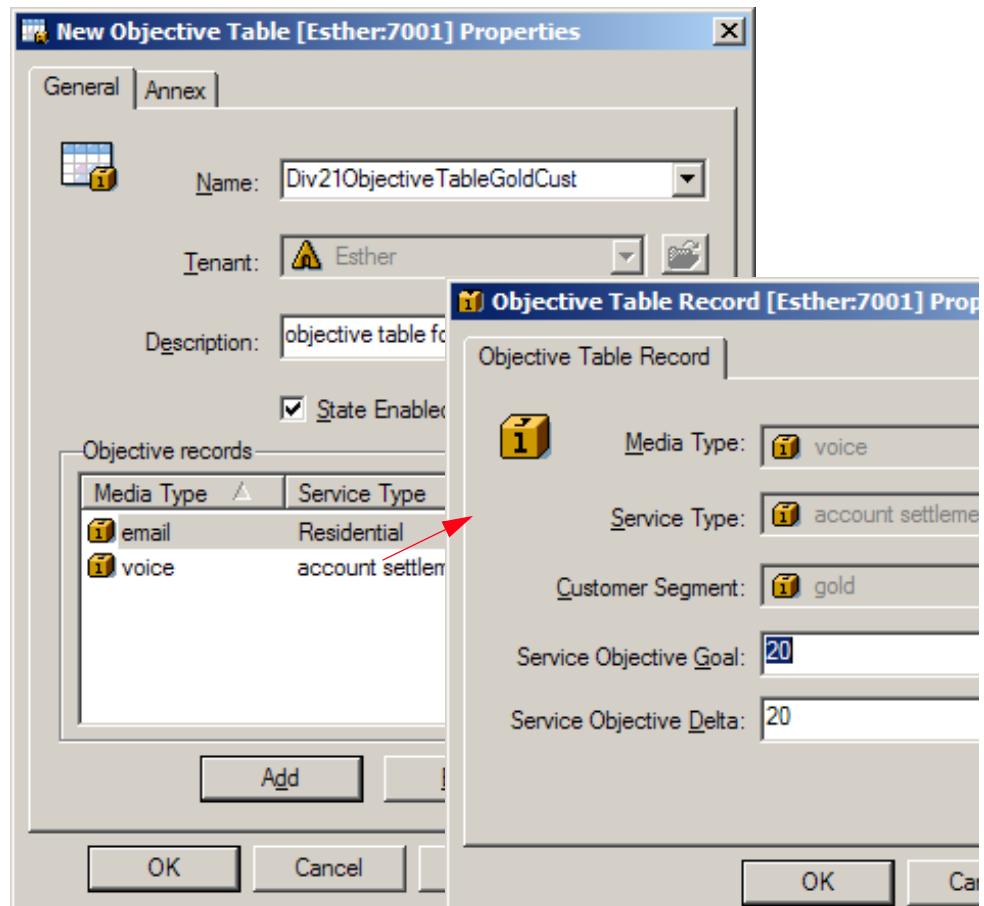


Figure 80: Objective Table Object in Configuration Manager

You then have the option of using the predefined baseline service objective or overriding it in the `MultiAttach Properties` dialog box (see Figure 78 on [page 141](#)).

In the example in Figure 80 on [page 145](#), if the customer calling in (voice Media Type) requests an account settlement Service Type (through an IVR via Caller Entered Digits attached by T-Server to the `EventRouteRequest`), and the customer belongs to the Gold Customer Segment (which can be determined by the strategy using a database lookup on the customer-entered account number), the service objective can then be used for routing.

For additional information on `Business Attributes` and service objectives, see the following functions:

- `GetCustomerSegment` (see [page 459](#))
- `GetServiceType` (see [page 460](#))
- `GetMediaType` (see [page 459](#))
- `FindServiceObjective` (see [page 480](#))



MultiFunction

The MultiFunction object allows you specify multiple functions in one IRD object and gives the option of writing the output to a variable. In contrast, the Function object (see [page 127](#)) allows only a single function.

You can use the MultiFunction object when implementing a Share Agent by Service Level Agreement (SLA) routing solution as described in the *Universal Routing 8.0 Routing Application Configuration Guide*. This type of routing solution enables a business user that manages multiple business lines to define the triggering conditions and constraints that allow agents to be shared among business lines.

SetTargetThreshold

As described in the above guide, one method for implementing an SLA solution involves associating borrowing and lending conditions (contained in an expression) with targets via strategy function SetTargetThreshold (see [page 612](#))

For example, assume that under certain conditions, agents serving VISA card customers can borrow agents from two other business lines, such as agents serving MC customers or agents serving DISCOVERY card customers. The conditions that must be met include both:

1. **Borrowing** conditions (example: when there are a certain number of interactions waiting in the queue for VISA card agents to handle) and
2. **Lending** conditions (example: conditions that will allow agents service the MC or DISCOVERY business lines to service VISA customers without negatively impacting their own business line).

By using the SetTargetThreshold twice in MultiFunction object, you can specify the borrowing/lending conditions for the two agent groups that VISA can potentially borrow from.

1. VISA's borrowing/lending conditions that apply to the MC agent group and
2. VISA's borrowing/lending conditions that apply to the DISCOVERY agent group.
3. [Figure 81](#) shows an example properties dialog box for the MultiFunction object that specifies the above two conditions.

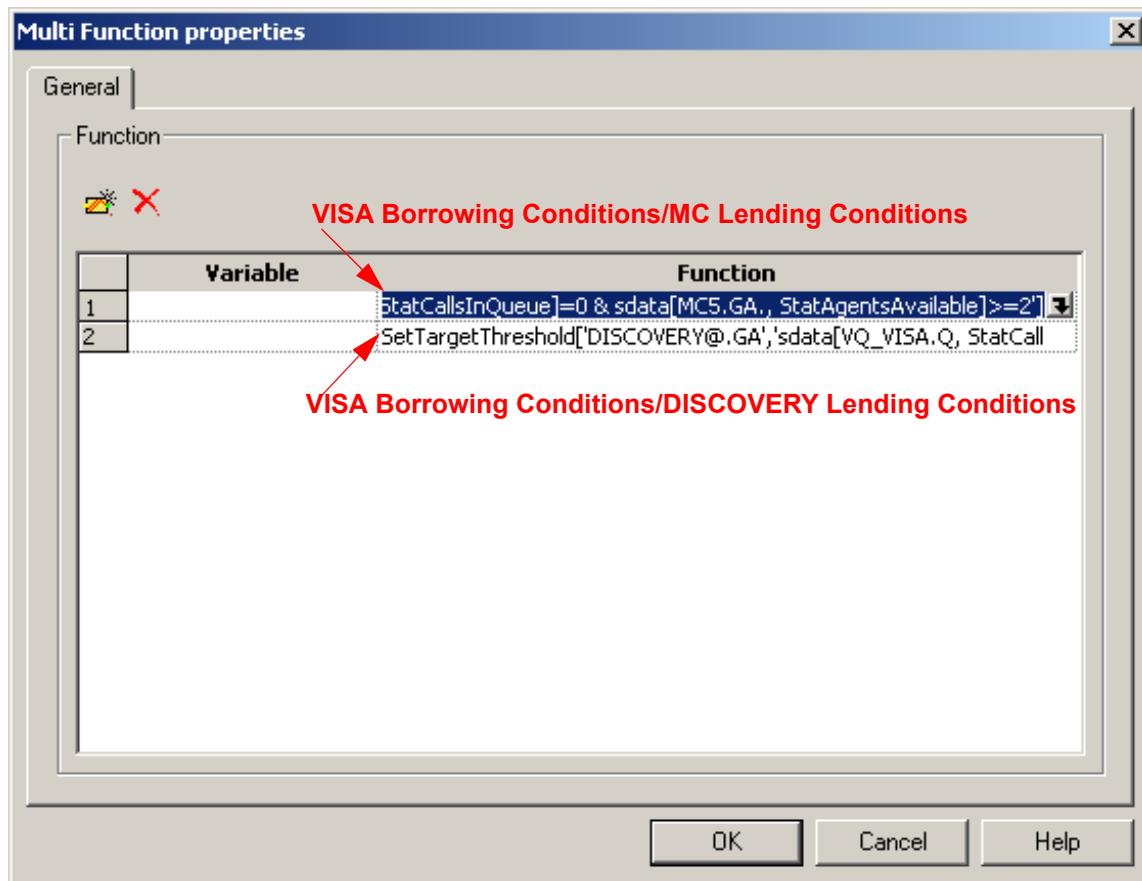


Figure 81: Multi Function Object Properties Dialog Box

In [Figure 81](#), note the following:

- You can write the output of the function to a variable (although in this example, `Variable` is not used).

Under `Function`, the `MultiFunction` properties dialog box shows the function and associated parameters previously entered in the `Function Properties` dialog box (see [Figure 82](#)), which opens when you click the down arrow in the `MultiFunction` Properties dialog box ([Figure 81](#) above).

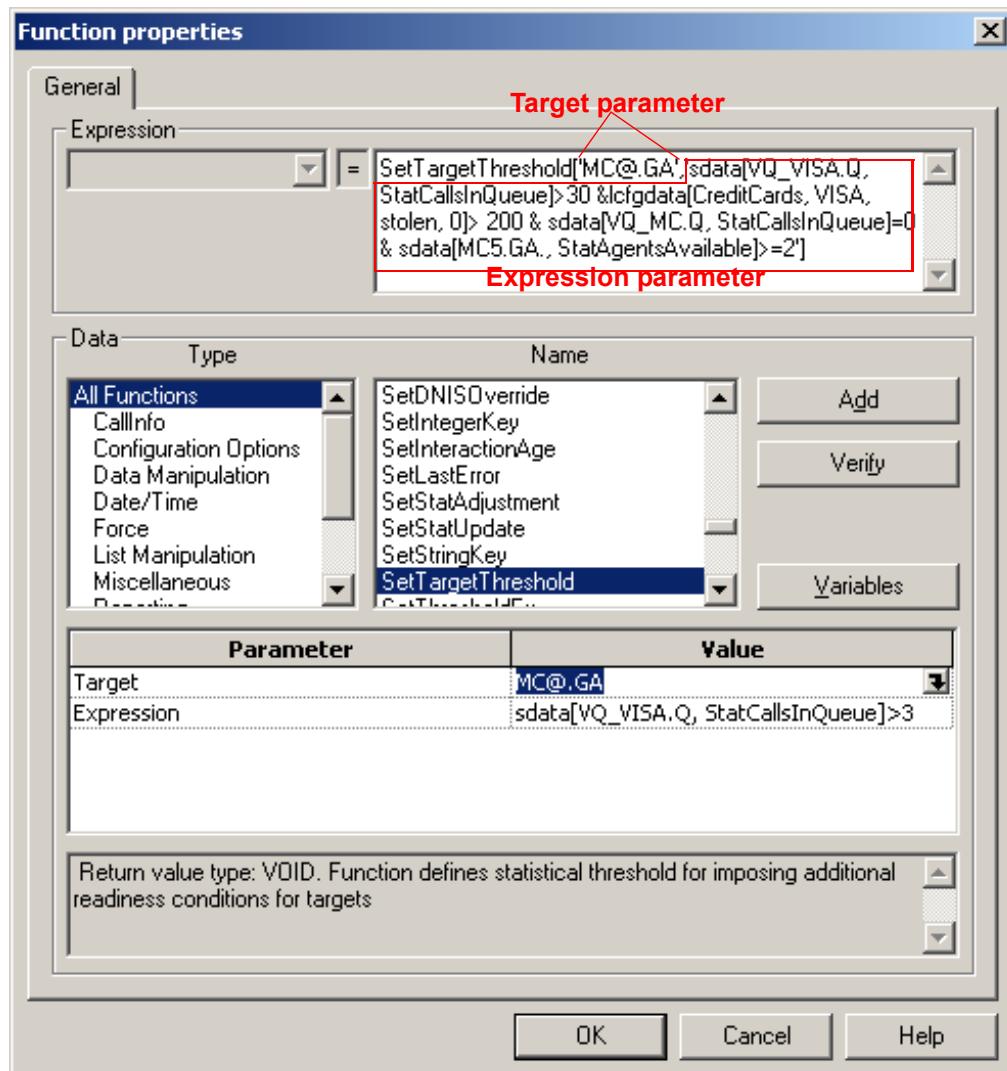


Figure 82: Function Properties Dialog Box

The top of the properties dialog box in [Figure 82](#) shows that function SetTargetThreshold was selected.

Note: Only if the expression is resolved to a not zero (0) value can the interaction be routed to the target.

For more information, see the chapter on Share Agent by Service Level Agreement in the *Universal Routing 8.0 Routing Application Configuration Guide*.

Multimedia Objects

You can use the Multimedia objects (see Figure 34 on [page 79](#)) when creating strategies that will be called by a business process.

Business Processes

A *business process*, executed by Genesys eServices's Interaction Server, is a series of strategies and queues connected in a logical way to form a workflow that directs the handling of eServices interactions.

An interaction is initially placed into an inbound queue. It is then taken out of the queue and submitted to a routing strategy. The routing strategy eventually routes the interaction to a target, but not necessarily the final target. The target processes the interaction and places it into another queue. This continues until processing is stopped or the interaction reaches some final (outbound) queue.

During its lifetime, an interaction may undergo various types of automated processing as well as handling by agents, supervisors, and QA. It may be submitted to different routing strategies and be pulled out of/placed into various queues a number of times. Developers create business process objects in IRD's Interaction Design window. [Figure 83](#) shows the start of an example business process in IRD's Interaction Design window.

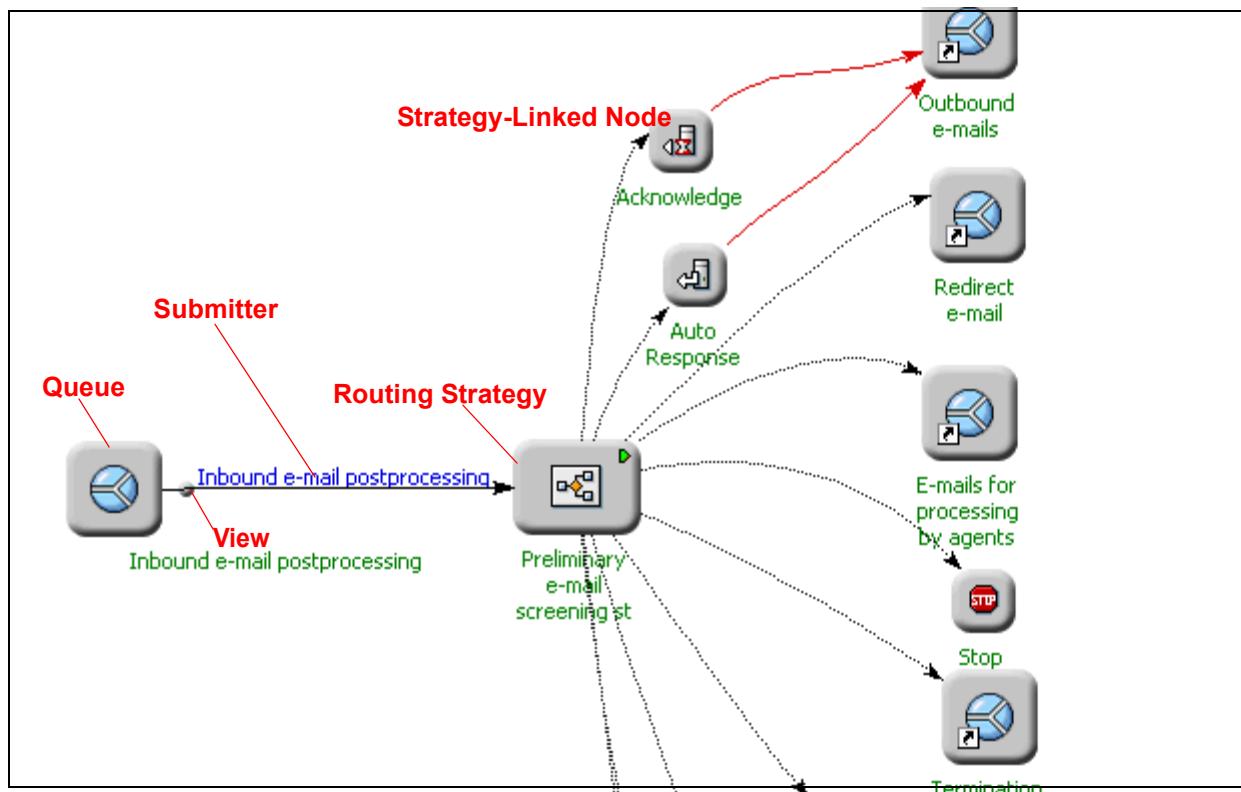


Figure 83: Example Business Process in IRD's Interaction Design Window

Note: For information on business process components, see the *Universal Routing Business Process User's Guide*.

In summary, routing strategies used in business processes function much like “stepping stones” between queues. A single business process can connect multiple routing strategies.

Interaction Design Objects

In addition to strategies used as single objects in the **Interaction Design** window, there are some other objects that you can create and use there. These include:

- Queues
- Media Servers
- Endpoints
- Submitters
- Views
- Workbins

For information about these objects, as well as how to use endpoints in the **Interaction Design** window to connect a media server to an inbound queue in an interaction workflow, see the **Interaction Workflow Design** section of the *Universal Routing 8.1 Interaction Routing Designer Help* and the *Universal Routing Business Process User's Guide*.

About Genesys Queues

Genesys software maintains both interaction and virtual queues.

- A queue in a business process executed by Interaction Server is called an *interaction queue* in this document. Business process developers use IRD’s **Interaction Design** window to create interaction queues (Queue objects). Once configured, interactions (such as e-mails and chats) are selected and extracted using View objects and submitted to Strategy objects that send them to another interaction queue or a target (such as an agent, workbin, or server). The definition of an interaction queue defined is stored as a **Script** object in the Configuration Database.
- All interactions submitted to URS are placed in *virtual queues*. A virtual queue is not a physical queue but rather a logical queue to which all interactions are queued if the specified targets in the Routing objects within a routing strategy are not available. See “Virtual Queues in Routing Objects” on [page 357](#) for more information.

Just as a business process keeps all its interactions in interaction queues, URS keeps all its interactions in virtual queues (its Connections list can have many different T-Servers). URS communicates with Interaction

Server about e-mail, chat, and open media types of interactions, and with voice T-Server about voice calls. Interactions sent to URS from Interaction Server are returned to Interaction Server if the target is unavailable.

- In addition to interaction and virtual queues, there are also ACDQueue routing target objects. However, the term *queue*, as used in this e-mail processing section, refers to an interaction queue.
- See also “Workbin” on [page 348](#).

Note: E-mail Server Java must specify an interaction queue’s Script name in order for Interaction Server to correctly handle an interaction. A queue’s Script name and its displayed (decorative) name in the Interaction Design window (see Figure 83 on [page 149](#)) may differ. Always use the value in the Interaction Queue properties dialog box, Script Name field. Open this dialog box by right-clicking the Queue object in the Interaction Design window and selecting Properties of. Otherwise, if the E-mail Server Java options default-inbound-queue or default-outbound-queue are set to the queue’s display name, Interaction Server can’t handle the interaction

URS Communication with Interaction Server

Using a subset of T-Server protocol, Interaction Server communicates with URS, which executes the logic of a business process consisting of various queues, views, and strategies. All messages from URS to Interaction Server start with Request. All messages from Interaction Server to URS start with Event. The basic message set is shown in [Table 14](#).

Table 14: Basic URS/Interaction Server

Name	Direction	Comment
EventRouteRequest	from Interaction Server to URS	Ask URS to find appropriate target (Place, Agent, ...) for given interaction
RequestRouteCall	from URS to Interaction Server	Provide Interaction Server with target information for a given interaction
EventRouteUsed	from Interaction Server to URS	Report to URS about successful processing or command to cancel strategy execution for this interaction (ReferenceID = 0)

Using a subset of T-Server protocol, Interaction Server communicates with URS when executing the logic of a business process, which consists of various queues and routing strategies.

RequestUpdateUserData, RequestAttachUserData, RequestDeletePair, RequestDeletePair, RequestDistributeEvent for EventQueued, EventDiverted, EventPartyAdded, EventPartyRemoved, and EventAbandoned are supported.

The following message types for third party servers (such as Universal Contact Server and E-mail Server Java) are also supported:

Table 15: URS/Interaction Server Request and Event Messages

Name or Request/Event	Direction	Comment
Request3rdServer	from URS to Interaction Server	Send to Interaction Server request for 3rd-party server operation
Event3rdServerResponse	from Interaction Server to URS	Returns results of 3rd-party server operation
Event3rdServerFault	from Interaction Server to URS	Supply error information about 3rd-party server operation failure

You can use these messages in the log file when diagnosing strategy or routing problems. For more information, see the [Genesys Events and Models Reference Manual](#). It provides examples of basic scenarios involving messaging between Interaction Server and other components, including URS.

User Data

Since Interaction Server uses T-Library protocol to communicate with URS, all of the interaction properties are presented to URS in a `UserData` property of the `T-Event` structure. Here is a sample of part of an `EventRouteRequest` message that provides interaction data to URS:

```
17:05:21.462 Trc 24102 Sending to router: universal_router_showtime:  
'EventRouteRequest' message:  
    AttributeExtensions [bstr] = TKVList:  
        'STRATEGY_ID' [str] = "strategy_first"  
        AttributeThisQueue [str] = "asl_strategy_first"  
        AttributeThisDN [str] = "asl_strategy_first"  
        AttributeCallType [int] = 0  
        AttributeCallID [int] = 3829  
        AttributeMediaType [int] = -1  
        AttributeConnID [long] =  
        AttributeCustomerID [str] = "Genesys"  
        AttributeUserData [bstr] = TKVList:  
            'AccountBalance' [int] = 0  
            'CustomNumber2' [int] = 0  
            'CustomNumber3' [int] = 0  
            'Priority' [int] = 0  
            'ServiceObjective' [int] = 0
```

```

'InteractionId' [str] = "tst00004e20"
'TenantId' [int] = 101
'MediaType' [str] = "email"
'InteractionType' [str] = "Inbound"
'InteractionSubtype' [str] = "InboundNew"
'InteractionState' [int] = 1
'IsOnline' [int] = 0
'IsLocked' [int] = 0
'Queue' [str] = "queue_first"
'SubmittedBy' [str] = "Workflow Engine Tester"
'ReceivedAt' [str] = "2004-04-14T22:10:35Z"
'SubmittedAt' [str] = "2004-04-14T22:10:34Z"
'DeliveredAt' [str] = "2004-04-26T23:30:02Z"
'PlacedInQueueAt' [str] = "2004-04-23T18:04:10Z"

```

Strategy-Building Objects For Business Processes

Strategy-building objects used inside strategies called by business processes are located:

- In the Data and Services object toolbar.
- In the Multimedia objects toolbar, as described in Table 16 on [page 156](#).
- In the Outbound objects toolbar (see Figure 152 on [page 294](#)).
- In the Segmentation objects toolbar (see Figure 188 on [page 374](#)). With the exception of the DNIS and ANI objects, all other Segmentation objects can be used for eServices routing strategies. Use the Classify and Screen Segmentation objects for text-based routing strategies only.
- In the Routing objects toolbar (see Figure 158 on [page 311](#)). Use only the Route Interaction, Queue Interaction, and Workbin objects for eServices routing strategies. Use all other objects for routing voice interactions.
- In the Miscellaneous objects toolbar (see Figure 60 on [page 120](#)), all objects can be used for eServices routing strategies.

The Voice Treatment objects (see Figure 213 on [page 405](#)) are used only in voice interactions.

For information on objects used outside of strategies in IRD's `Interaction` Design window, such as the Media Server object, see the Interaction Workflow section of the *Universal Routing 8.1 Interaction Designer Help*.

Strategy-Linked Nodes

When a routing strategy is used in a business process (see [page 149](#)), certain IRD objects used in the strategy cause *strategy-linked nodes* to automatically appear in the workflow (see Figure 83 on [page 149](#)).

The following example shows how strategy-linked nodes automatically appear in a business process:

1. An e-mail routing strategy is using the Route Interaction object (see Figure 84):

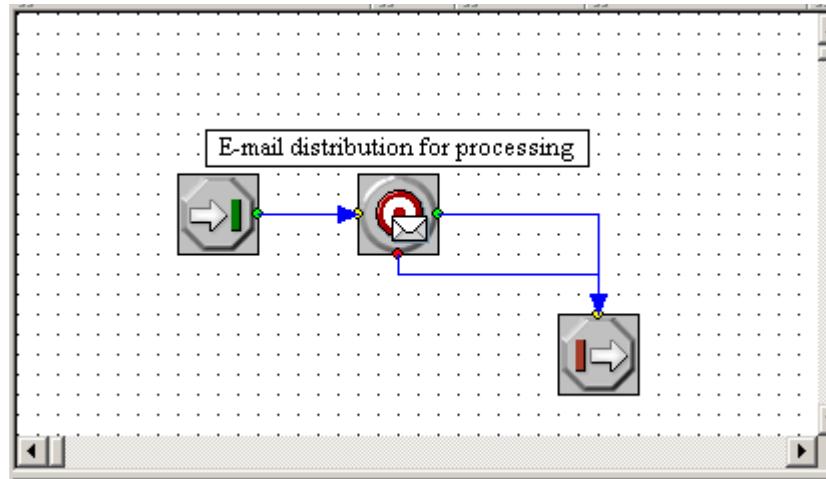


Figure 84: Example Routing Strategy With Route Interaction Object

2. The Interaction Queue tab in the Route Interaction Properties dialog box appears as follows (see Figure 85):

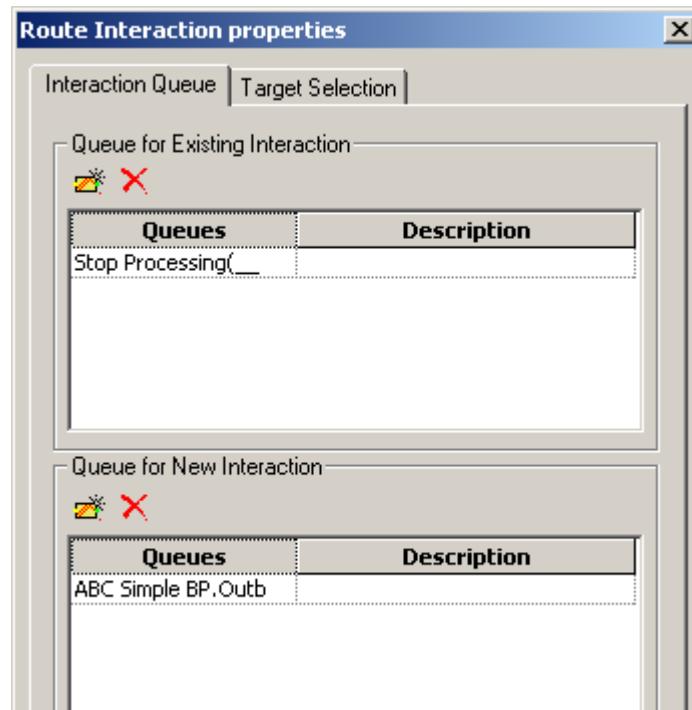


Figure 85: Interaction Queue Tab for Route Interaction Object

3. The Target Selection tab in the Route Interaction Properties dialog box appears as follows (see Figure 86):

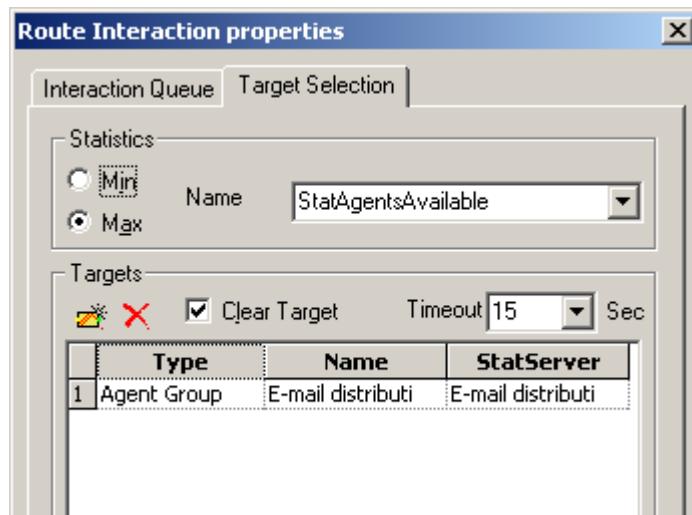


Figure 86: Target Selection Tab for Route Interaction Object

When you use the Process ABC e-mail routing strategy shown in Figure 84 on [page 154](#) in a business process, agent target, queue, and stop nodes automatically flow out the right side of the strategy (see [Figure 87](#)).

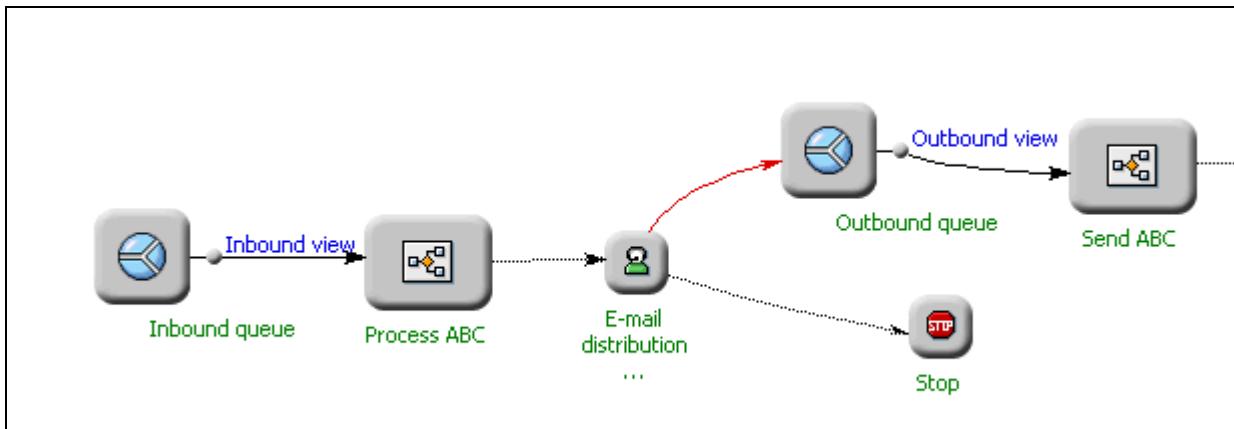


Figure 87: Strategy-Linked Nodes in a Business Process

- The agent strategy-linked node (E-mail distribution) reflects the agent group specified in the Target Selection tab of the Route Interaction object (see [Figure 86 on page 155](#)).
- The queue strategy-linked node (Outbound queue) reflects the queue specified under Queue For New Interaction in the Interaction Queue tab of the Route Interaction object (see [Figure 85 on page 154](#)).
- The stop strategy-linked node (Stop) reflects the Stop Processing(_ entry under Queue for Existing Interaction in the Interaction Queue tab of the Route Interaction object (see [Figure 85 on page 154](#)). A stop node also appears in a business process when the Stop Interaction object (see [page 274](#)) is used in a strategy.

[Table 16](#) indicates if strategy-linked nodes appear for a Multimedia object.

Summary of Multimedia Objects

[Table 16](#) summarizes the Multimedia objects in alphabetical order, as well as some additional objects used for eServices interaction processing. Detailed descriptions of each object follow in the pages ahead.

Table 16: Summary of Multimedia Objects

Name	Purpose	Strategy-Linked Nodes
Acknowledgement (page 161)	<p>Use to generate (but not send) an acknowledgement e-mail using text either from your Standard Response library or from User Data explaining that the customer's message (in any text-based format, including fax or SMS) has been received and will be answered by an agent.</p> <p>The Acknowledgement object has two methods for getting text for the acknowledgement e-mail.</p> <ol style="list-style-type: none"> 1. URS automatically selects Standard Response text based upon the category specified by the Classify object and placed in the interaction's User Data. 2. You can select a Standard Response. You are presented with a tree of categories and associated Standard Responses that have been configured in Configuration Manager. 	When Operation Mode is E-Mail or Open Media (see Figure 88 on page 162), generates server and queue nodes in a business process.
Attach Categories (page 178)	Enables you to manually emulate the results of the Classify object's request to Classification Server to attach classification categories to an interaction.	
Autoresponse (page 181)	<p>Use to generate (but not send) an e-mail with Standard Response text. Use the Send E-Mail object to send this message after it has been created.</p> <p>The Autoresponse object uses one of two methods for selecting a Standard Response.</p> <ol style="list-style-type: none"> 1. URS automatically selects Standard Response text based upon the category specified by the Classify object and placed in the interaction's User Data. 2. You can select a Standard Response. You are presented with a tree of categories and associated Standard Responses that appear in Configuration Manager. 	When Operation Mode is E-Mail or Open Media (see Figure 105 on page 183), generates server and queue nodes in a business process.

Table 16: Summary of Multimedia Objects (Continued)

Name	Purpose	Strategy-Linked Nodes
Chat Transcript (page 191)	<p>Use to generate (but not send) an e-mail with the customer's chat transcript attached using text from your Standard Response library.</p> <p>Use the Send E-Mail object to send this message after it has been created.</p>	Generates server and queue nodes in a business process (see Figure 105 on page 183).
Classify (page 198) Note: There is also a Classify Segmentation object (page 377)	<p>Use to request Classification Server to assign one or more categories to an interaction. The categories can then be used to select a Standard Response, as well as for routing or reporting.</p> <p>Requires that you specify a Classification Server.</p> <p>The input parameters include the minimum relevancy each category must have in order for Classification Server to consider an interaction as belonging to that category (threshold). Parameters also specify whether to analyze only for the parent categories set up in Configuration Manager or both parent and child categories.</p>	
Create Interaction (page 205)	Use to have URS request that an interaction record be created in the UCS Database. If the customer contact associated with the interaction is not found, this object can create new contact in the database.	
Create Email Out (page 212)	<p>Use to create a new outbound e-mail from a strategy. This outbound e-mail is sent via an E-mail Server Java, that has been specifically configured to handle Open Media interactions.</p> <p>Use the Send E-Mail object to send this message after it has been created.</p>	Generates server and queue nodes in a business process.
Create Notification (page 217)	<p>Use to create a new outbound e-mail from a strategy. This e-mail can contain information requested by a customer or can direct the customer to a URL where the information is located.</p> <p>This outbound e-mail is sent via an E-mail Server Java, that has been specifically configured to handle Open Media interactions.</p> <p>Use the Send E-Mail object to send this message after it has been created.</p>	Generates server and queue nodes in a business process.

Table 16: Summary of Multimedia Objects (Continued)

Name	Purpose	Strategy-Linked Nodes
Create SMS (page 221)	<p>Use to create a new outbound e-mail message in a format designed to be sent by a specially configured E-Mail Server Java to SMS-ready targets such as cell phones via an E-Mail-to-SMS gateway.</p> <p>Use the Send E-Mail object to send this message after it has been created.</p>	Generates server and queue nodes in a business process.
Distribute Custom Event (page 226)	Use to define and distribute a custom reporting event to an Interaction Server. This object causes URS to generate a request to the Interaction Server for the method <code>DistributeCustomEvent</code> .	
Find Interaction (page 228)	Use the Find Interaction object to query Interaction Server for all or a number of interactions that correspond to specific conditions and listed in a specific order.	
Forward E-Mail (page 233)	<p>Use to generate (but not send) an incoming message to an external resource with the expectation of getting a response back. Allows you to select a Standard Response.</p> <p>Use the Send E-Mail object to send this message after it has been created.</p>	Generates server and queue nodes in a business process.
Identify Contact (page 237)	<p>Use to locate one or more contacts in the UCS Database with attributes that match data attached to the interaction. If you choose to match one unique contact, if UCS locates it, it returns all stored data concerning that contact.</p> <p>You can also choose to have UCS create a new contact if no existing contact matches the attached data.</p> <p>This object requires you to specify a Contact Server.</p>	
MultiScreen (page 241)	<p>Use to request Classification Server to screen text-based interactions for certain words or patterns of words using multiple rules set up in Knowledge Manager.</p> <p>You can choose to return Screening Rule identifiers (associated with categories in Knowledge Manager), or categories (associated with Standard Responses in Knowledge Manager), matched pairs, or all of the above.</p> <p>Requires that you name the Classification Server used for screening as well as a language and a Screening Rule.</p>	

Table 16: Summary of Multimedia Objects (Continued)

Name	Purpose	Strategy-Linked Nodes
Queue Interaction (page 320) (this object is located in the Routing objects toolbar)	Generates a message from URS to Interaction Server to place the interaction back into an interaction queue. Requires you to specify an interaction queue. Queues configured in the Interaction Design window are listed for selection.	
Redirect E-Mail (page 250)	Use to generate (but not send) an incoming message to an external resource (which can be another agent such as an “expert” or an e-mail server) without expecting a response or any further processing. Use the Send E-Mail object to send this message after it has been created.	Generates server and queue nodes in a business process.
Render Message Content (page 254)	Use this object to create a message content using text from either a message box of the strategy object, strategy variable, User Data or Standard Response.	
Reply E-Mail from External Resource (page 258)	Takes an External Resource Reply inbound e-mail as input, extracts the external resource reply text from it, creates a Customer Reply outbound e-mail, and submits the e-mail to Interaction Server.	Generates server and queue nodes in a business process.
Route Interaction (page 322) (This object is located in the Routing objects toolbar)	Routes an interaction to a specified target type (Agent, AgentGroup, Campaign Group, Place, PlaceGroup, Variable or Skill expression). You can select interaction queues for Existing and New interactions (optional). The interaction queues that have been configured in the Interaction Design window appear for selection. Send to a New interaction queue as an escalation mechanism in order to increase interaction priority in the queue. You can also select the Stop Processing queue. This queue is predefined in Configuration Manager.	Generates a queue node in a business process. If any type of target is specified generates a target node in a business process.
Screen (page 263)	Use to request Classification Server to screen text-based interactions for certain words or patterns of words using a Screening Rule set up in Knowledge Manager. Requires that you specify the Classification Server used for screening as well as a language and a Screening Rule.	

Table 16: Summary of Multimedia Objects (Continued)

Name	Purpose	Strategy-Linked Nodes
Send E-Mail (page 269)	<p>Use to send an e-mail that is waiting in an interaction queue to a customer or another agent. This e-mail can be the final e-mail response or an e-mail generated with the Acknowledgement, Autoresponse, Chat Transcript, Forward E-Mail, Create Email Out, Create Notification, or Redirect E-Mail object.</p> <p>You can specify an E-mail Server Java to send the e-mail.</p>	
Stop Interaction (page 274)	<p>Sends a stop interaction request to Interaction Server. Requires a Reason Code.</p> <p> Optionally, you can also have it notify UCS or a third party server (through Interaction Server) that processing has stopped for a particular strategy within a business process.</p> <p>If you select the Delete check box, the interaction will be deleted from the UCS Database.</p>	Generates a stop node in a business process.
Submit New Interaction (page 277)	Use to define and submit a new interaction to an Interaction Server. This object causes URS to generate a request to the Interaction Server for the method <code>SubmitNew</code> .	
Update Contact (page 281)	<p>Use to create new contact attributes or to update existing ones.</p> <p>Requires that you specify a Contact Server and (optionally) parameters. The <code>ContactId</code> must exist as attached data for the Update Contact object to work.</p>	
Update Interaction (page 285)	Use this object to update properties of the interaction in the Interaction Server database that are not currently processed in the strategy by URS.	
Update UCS Record (page 290)	Use this object to update properties of the interaction in the Universal Contact Server database that are not currently processed in the strategy by URS.	
Workbin (page 348) (This object is located in the Routing objects toolbar)	Use to route an interaction to a storage area associated with an Agent, Agent Group, Place or Place Group. For example, an Agent Workbin may hold draft e-mail responses. Workbins are created in the Interaction Design window.	

Finishing a Routing Strategy

Always finish a non-voice routing strategy used in a business process with one of the following routing objects: Stop Interaction (see [page 274](#)), Queue Interaction (see [page 320](#)), Route Interaction (see [page 322](#)) or Workbin (see [page 348](#)).

This section now continues with a discussion of each Multimedia object.



Acknowledgement

Use the Acknowledgement object to request E-mail Server Java to use text from your Standard Response library to generate (but not send) an acknowledgement e-mail indicating that the customer's message has been received. The Standard Response may also indicate that the customer's message will be answered by an agent.

Different Acknowledgement receipts can be used in different situations. For example, one receipt can be written for nights and weekends; another can be written for holidays; another can be written when there is a long queue of interactions to be handled (including when there is a backlog of messages in the media server).

Note: You should not use an Acknowledgement object if you already used an Autoresponse object to create a response to the customer's message.

Configuring the Acknowledgement General Tab

The Acknowledgement object has three operation modes: Email, Open Media, and 6.x Compatible (see “6.x Compatible Operation Mode” on [page 176](#)).

[Figure 88](#) shows the General tab of the Acknowledgement Receipt Properties dialog box with the Open Media mode selected.

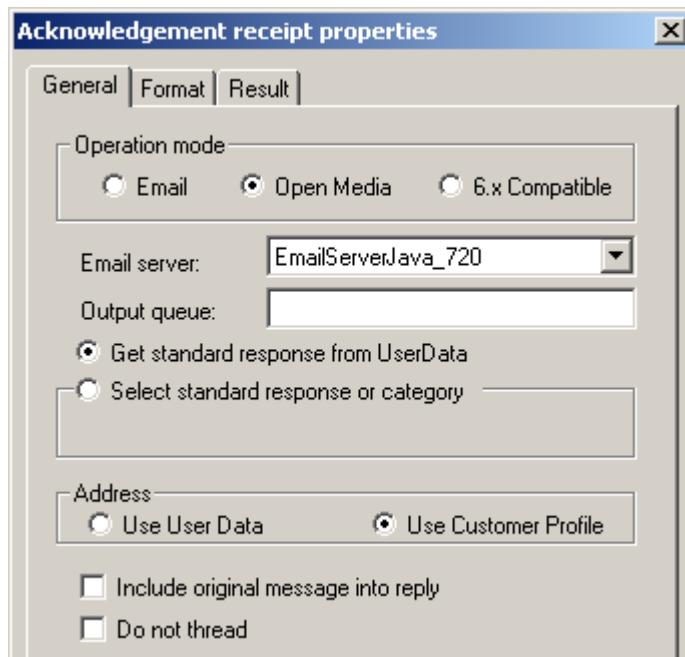


Figure 88: Acknowledgement Object, General Tab

The Email and Open Media modes use a tree viewer to represent pre-written Standard Responses (see [Figure 89](#)).

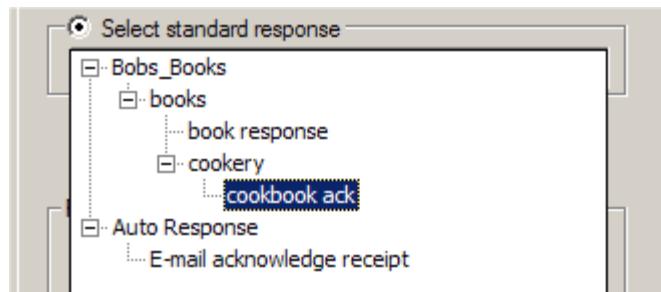


Figure 89: Tree Viewer for Acknowledgement Standard Responses

Standard response display names carry over from Knowledge Manager. These names appear in Configuration Manager in the Business Attributes > Category Structure folder (see [Figure 90](#)).

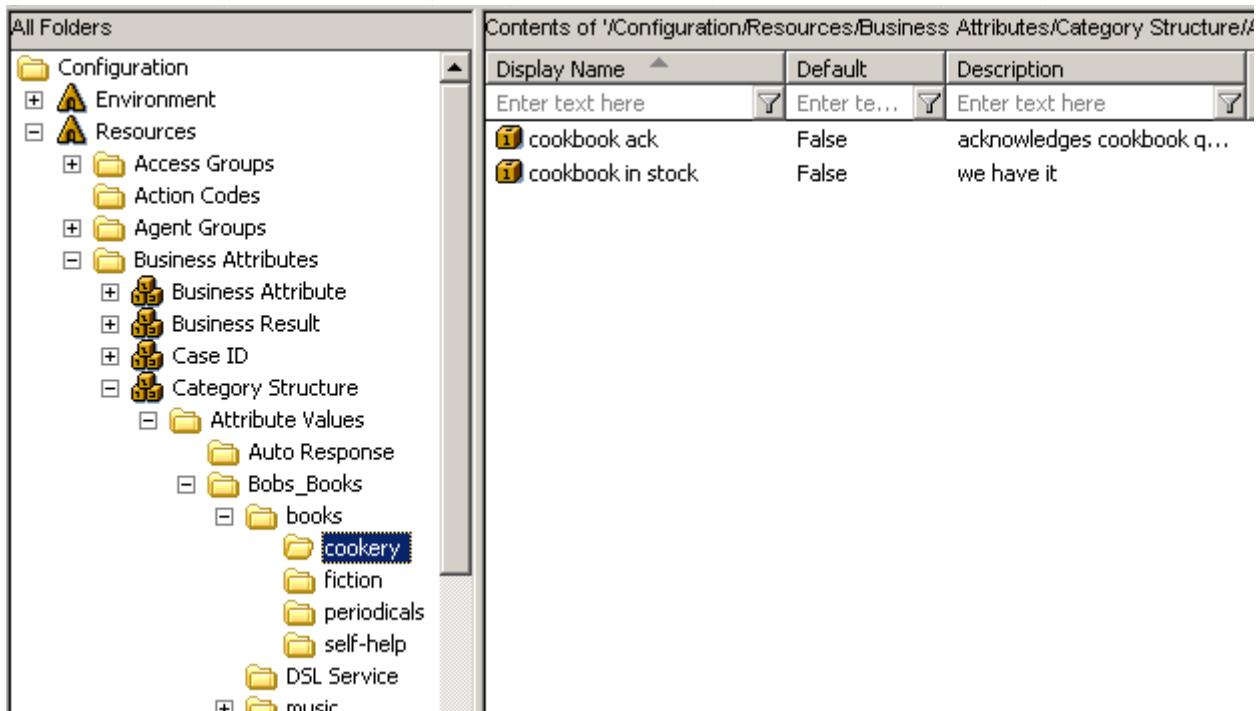


Figure 90: Category Structure Folder in Configuration Manager

Strategy-Linked Nodes from an Acknowledgement Object

When a routing strategy in a business process uses the Acknowledgement object, the specified E-mail Server Java and interaction Output queue (see Figure 88 on [page 162](#)) for the acknowledgement e-mail appear as strategy-linked nodes (as shown in the Acknowledge and Outbound e-mails objects in Figure 91).

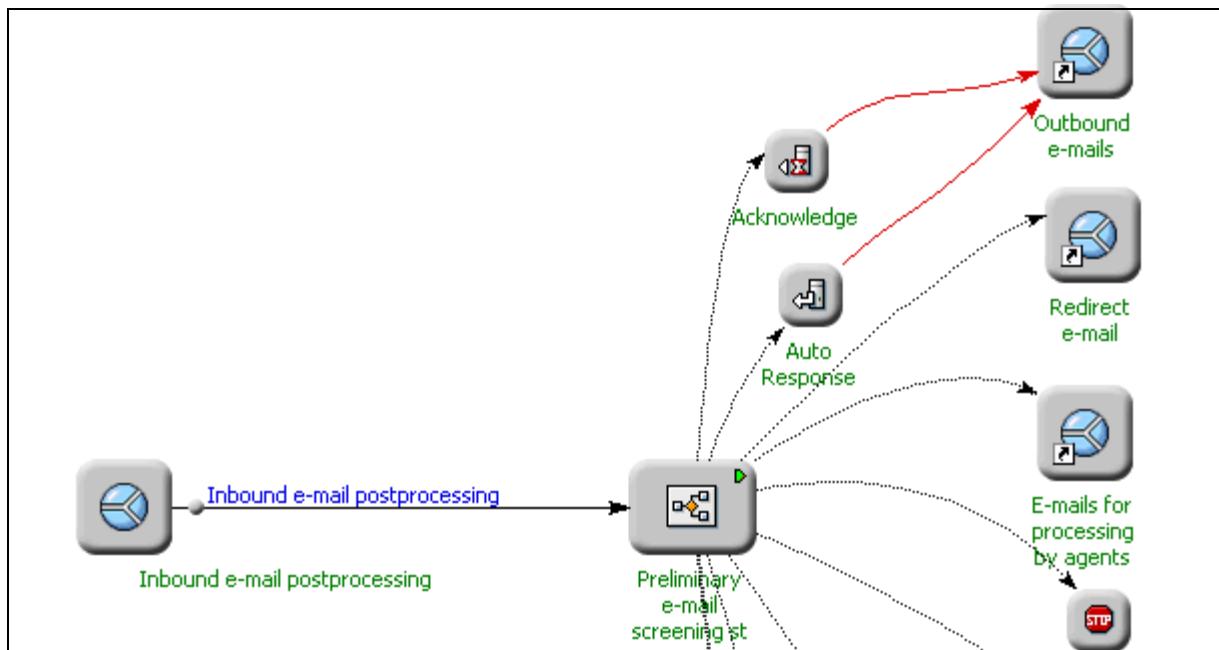


Figure 91: E-mail Server Java and Output Queue in a Business Process

Use the information in [Table 17](#) to complete the fields in the General tab of the Acknowledgement Receipt Properties dialog box in Email or Open Media mode.

Table 17: Acknowledgement Properties

Parameter	Description
Operation mode E-Mail	Select if this acknowledgement will be sent by a standard instance of E-Mail Server Java.
Operation mode Open Media	Select if this acknowledgement will be sent by an E-Mail Server Java modified to handle Open Media interactions.
E-mail Server	Click the down arrow and select the Application name in Configuration Manager for the E-mail Server Java to be notified. <ul style="list-style-type: none"> You can leave this field empty. If empty, Interaction Server uses the first available E-mail Server Java named in its Connections list.
Output queue	Required. Click in the Output queue text box to expand the business processes. Select the interaction queue where E-mail Server Java should place the acknowledgement e-mail.

Table 17: Acknowledgement Properties (Continued)

Parameter	Description
Get standard response from UserData	<p>If this button is clicked on, the Select Standard Response button is disabled and vice-versa.</p> <p>Uses the classification codes returned by the Classify object and stored in the interaction UserData to get a Standard Response.</p> <ul style="list-style-type: none"> • To use this field, the following key must be present in the interaction User Data: <code>CategoryId</code>. If <code>SRLId</code> and <code>CtgId</code> in the Request message and <code>CategoryId</code> in UserData are null, then an error is generated and the interaction goes out the Error port. • Note: If you select this option instead of the alternative, Select Standard Response from User Data, you cannot use the Field codes functionality described below.
Select standard response or category	<p>Use this option to select a Standard Response (such as cookery in Figure 90 on page 163) or category (such as Bobs_Books or books in Figure 90 on page 163) to indicate the pre-written text to be included in the acknowledgement e-mail to the customer. Standard responses are initially defined in Knowledge Manager, stored in the UCS Database, then carried over to Configuration Manager. For more information on Standard Responses, see the <i>eServices (Multimedia) 8.1 User's Guide</i>.</p> <p>Notes:</p> <ul style="list-style-type: none"> • If you select this option, you can use the Field codes functionality described below (if the selected Standard Response contains one or more field code custom variables).
Address (Open Media mode only)	<p>Select whether to take the address to which the acknowledgement should be sent from user data attached to the interaction or from the customer's profile stored in the UCS Database.</p> <p>If you choose to take the address from the database, you must place the Identify Contact object in the strategy before the Acknowledgement object. For information on the Identify Contact object, see "Identify Contact" on page 237.</p> <p>Note: This pane does not appear in E-Mail mode.</p>
Include original message into reply	Select the check box if the Acknowledgement e-mail should attach the original message.

Table 17: Acknowledgement Properties (Continued)

Parameter	Description
Do not thread	If this box is checked, it creates a new entry in the UCS Database that is not part of the original thread.
Field codes	<p>You can use this pane to assign values to any Field Code Custom Variables that may have been associated with the Standard Response in Knowledge Manager. See the <i>eServices (Multimedia) 8.1 User's Guide</i> for more information on field codes.</p> <p>To assign field code values, click the new item button (). A new row appears under the Key and Value headings. Under Key, existing Field Code Custom Variables associated with the response appear for selection. Under Value, any existing strategy variables display for selection. Use the information in the “Field Codes and Custom Variables” section below to edit the values.</p> <p>Note: You can also manually enter field code keys and values.</p>

Field Codes and Custom Variables

Both field codes and custom variables are defined in Knowledge Manager, as described in the *eServices (Multimedia) 8.1 User's Guide*.

Field codes are used to personalize Standard Responses. For example, your Standard Response might read:

Dear <\$Contact.FirstName\$>,

<\$Agent.Signature\$>

There are two field codes in this response. When an agent inserts this response into an e-mail, the first field code, <\$Contact.FirstName\$>, is replaced by the contact's first name as it appears in the UCS Database. The second field code, <\$Agent.Signature\$>, is replaced by the agent's signature as it appears in Configuration Manager.

[Figure 92](#) shows the Knowledge Manager Edit Standard Response dialog box with a Standard Response that uses field codes.

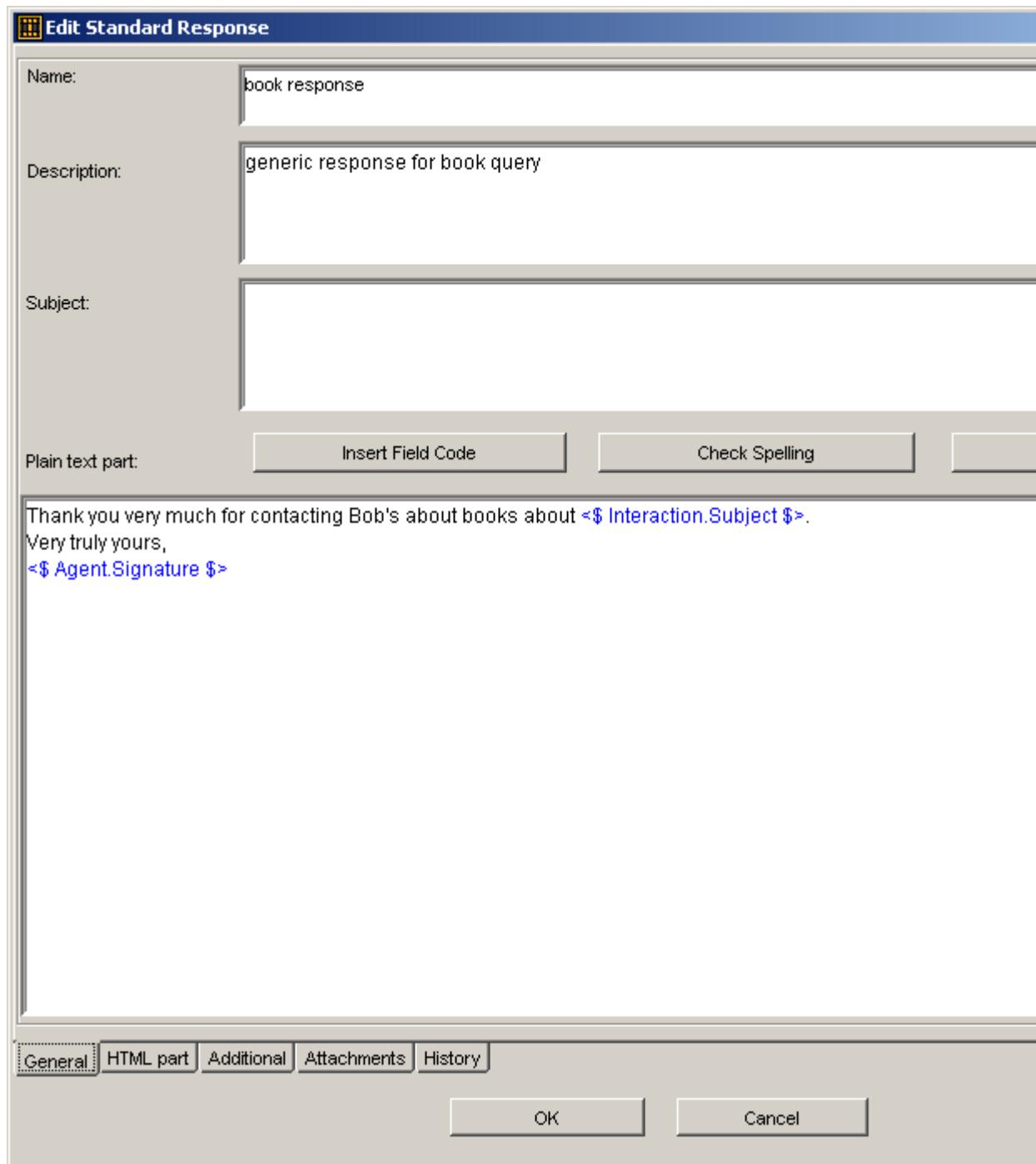


Figure 92: Knowledge Manager Standard Response With Field Codes

Field codes can also consist of custom variables, which can further particularize a Standard Response.

The Knowledge Manager `Edit Field Code` dialog box is where you define custom variable names. The dialog box shown in [Figure 93](#) has one custom variable defined, `Query_Topic`.

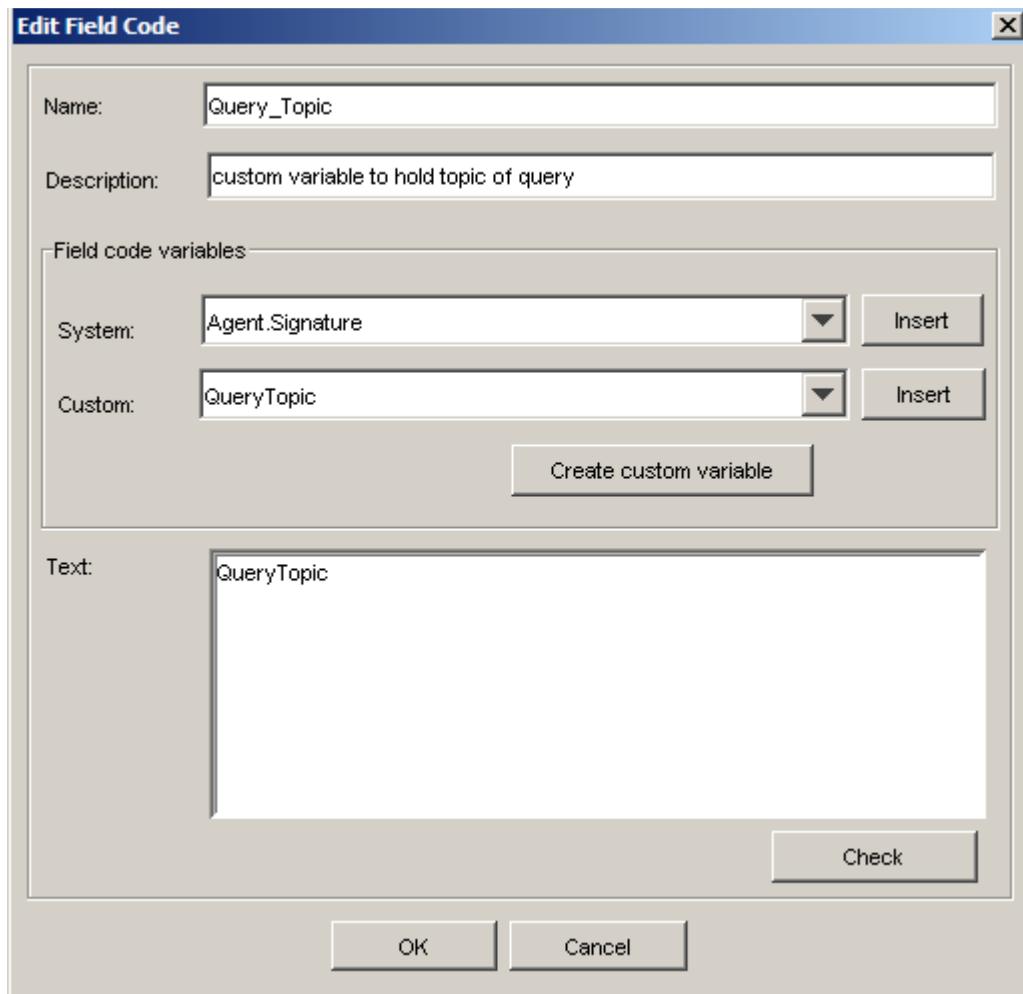


Figure 93: Knowledge Manager Edit Field Code Dialog Box

You can use the Field codes area in the E-mail Acknowledgement Receipt Properties dialog box (see Figure 88 on [page 162](#)) to further particularize a response by assigning strings to custom variables associated with field codes in Knowledge Manager.

Example of Custom Variable Usage

In Knowledge Manager, you configured a custom variable called `WesternAgentGroup`. You use this custom variable in a field code and then use that field code in a Standard Response that is subsequently selected in the General tab of the Acknowledgement object.

To use it:

- In the Acknowledgement object, enter `WesternAgentGroup` under Key and `Hello you have reached the western agent group. Welcome.` under Value in the Field codes pane (see Figure 88 on [page 162](#)).

The `WesternAgentGroup` variable is replaced by the string `Hello you have reached the western agent group. Welcome.` in the acknowledgement e-mail message.

Configuring the Acknowledgement Format Tab

You can customize the To, From, and Subject areas in the e-mail to the customer.

Note: The Acknowledgement, Autoresponse, and Reply E-Mail From External Resource objects use the same Format tab.

Figure 94 shows the Format tab in the Acknowledgement Receipt Properties dialog box with Custom selected.

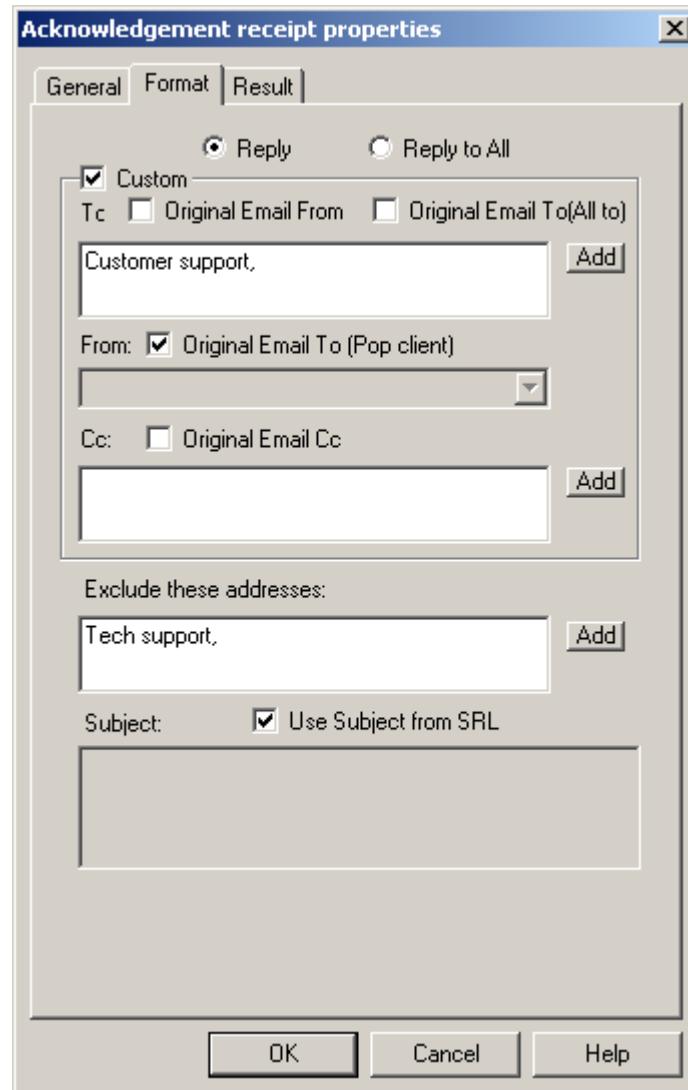


Figure 94: Acknowledgement Object Format Tab

Use the information in [Table 18](#) to complete the fields in the Format tab.

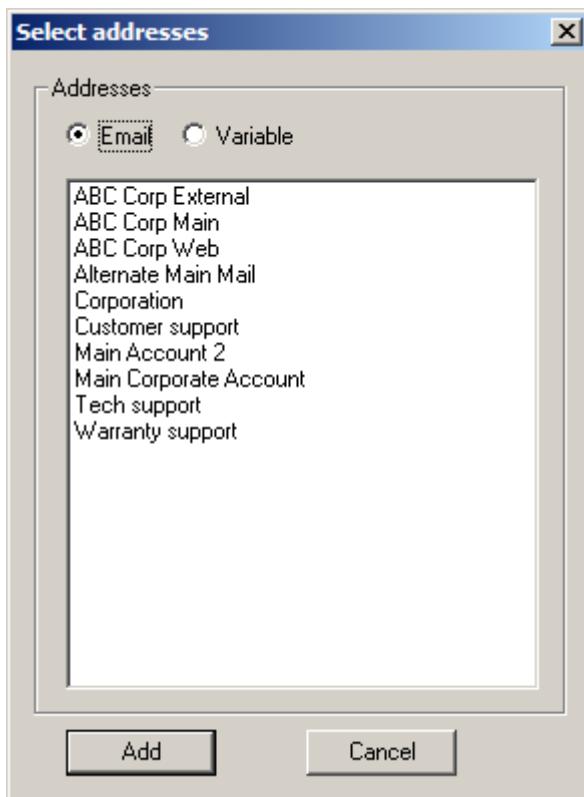
Table 18: Format Tab

Parameter	Description
Reply Reply to All	Select to indicate whether the acknowledgement should be sent to all addresses contained in the original message (Reply to All) or just the sender (Reply).
Custom	If applicable, select to add addresses to the acknowledgement. If you select Custom, the first two Add buttons on the right side of the dialog box are enabled. Note: When generating the acknowledgement, E-mail Server Java overrides the default-from-address setting configured in the server's Application object in the following circumstances: <ul style="list-style-type: none"> • If the Custom check box in the Format tab is cleared. • If the Custom check box in the Format tab is selected, but the From drop-down box is empty and the From check box is cleared.
To: Original E-mail From	If applicable, select to add addresses contained in the original message From area to the To area in the acknowledgement. This action enables the first Add button, which you must click to open a dialog box for selection of addresses (see Figure 95). You can select from a list or you can use the content of a variable.
To: Original E-mail To (All to)	Select to add addresses contained in the original message To area to the To area in the acknowledgement. This action enables the second Add button, which you must click to open a dialog box for selection of addresses (see Figure 95). You can select from a list or use the content of a variable.
Add	Click to add addresses. This action opens the dialog box shown in Figure 95 where you can add addresses.
From: Original E-mail To Pop Client	Clear this check box to select an address for the From area of the acknowledgement. Addresses in the drop-down list were previously entered as E-mail Accounts under Business Attributes in Configuration Manager (see Figure 48 on page 102).
Cc: Original E-mail Cc	Select to add addresses contained in the original message Cc area to the Cc area of the acknowledgement. This action enables the second Add button, which you must click to open a dialog box for selection of addresses to which the acknowledgement will be copied (see Figure 95). You can select from a list or use the content of a variable.

Table 18: Format Tab (Continued)

Parameter	Description
Exclude these addresses:	Enter addresses to exclude or click Add to select addresses to exclude (see Figure 95). Can contain a comma-separated list of domains, aliases, variables, and e-mail addresses.
Subject Use from SRL	Select to have the acknowledgement use the Subject from the Standard Response Library response. Clear the checkbox to enter a subject in the Subject text box.

[Figure 95](#) shows the dialog box that opens when you click the Add button in the Format tab.

**Figure 95: Select Addresses Dialog Box**

Use the information in [Table 19](#) to complete the Select Addresses dialog box.

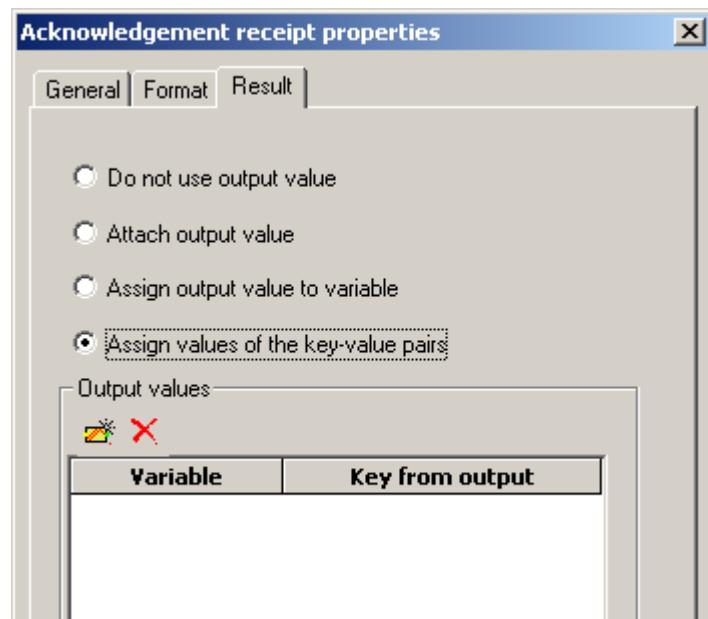
Table 19: Select Addresses Dialog Box

Parameter	Description
E-mail	Click the E-mail radio button and select one or more addresses or domain names from the list. All the addresses and domain names are configured as E-mail Accounts Business Attributes in Configuration Manager (see Figure 48 on page 102).
Variable	Select the variable radio button to select from available variables.
Add	Click Add to close the dialog box and return to the Format tab (see Figure 94 on page 169).

Configuring the Result Tab

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Figure 96 shows the Result tab in the Acknowledgement Receipt Properties dialog box with Assign values of the key-value pairs selected.

**Figure 96: Acknowledgement Object, Result Tab**

Use the information in [Table 20](#) to complete the Result tab (see Figure 96 on [page 172](#)) in the Acknowledgement Receipt Properties dialog box.

Table 20: Acknowledgement Result Tab

Parameter	Description
Do not use output value	Select this option to have URS disregard parameters data received from the server.
Attach output value	Selected by default. Select to attach returned values to the interaction User Data in the form of key-value pairs.
Assign output value to a variable	Select to assign returned output values to a variable. The variable must already be created (see “Variables in Objects and Expressions” on page 92). URS creates a string composed of the parameters data in the form of key1:value1 key2:value2 keyN:valueN and assigns this string to the selected variable.
Assign values of the key-value pair	Select to assign values in the returned array to a list of variables (see section on Returned Results). If you select this option, the Result tab activates the Output Values pane shown in Figure 96 .
Output values	Click the add item button () and select a variable and the key from the output to indicate the value to assign to the variable.

Returned Results for the Acknowledgement Object

The results returned will be either an Event3rdServerResponse message or an Event3rdServerFault message (see [Table 22](#)).

The content of the Event3rdServerResponse message is shown in [Table 21](#):

Table 21: Acknowledgement Object Returned Results

Key	Description
NewInteractionId	The interaction ID of the new outbound e-mail.
SrlId	Global unique Standard Response identifier if it was found and used.

Fault messages use the fault codes shown in Table 22 on [page 174](#).

Table 22: Acknowledgement Object Fault Codes

FaultCode value	FaultString value	Description	Action
105	No text found in Standard Response <SR_name>	The Standard Response exists but it contains no text	Exit from strategy
106	No ‘From/To’ address found	Either the From or the To address is missing	Exit from strategy
107	Too many ‘From/To’ addresses found	Either the From or the To field (or both) contain too many addresses	Exit from strategy
108	‘incorrect name’. Invalid ‘address’	The address or the domain is invalid	Exit from strategy
109	Standard response <SR_name> validity period ‘not yet started/expired’	The period during which the specified Standard Response is valid has not started or has ended	Exit from strategy
111	Standard response <SR_name> not found	This Standard Response does not exist in the database.	Exit from strategy
112	No Standard Response found in database	The database has no Standard Responses in it	Exit from strategy
201	Missing parameter name	One parameter is missing	Exit from strategy
202	Parameter ‘1’ and ‘2’ are not allowed	These two parameters cannot be present simultaneously	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
204	Incorrect value for parameter <parameter_name>, expected value 1 but was value 2	The specified parameter has an invalid value	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected message from the server’s client	End of processing

Table 22: Acknowledgement Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
510	Object not found in database	Object is not found in database	End of processing
513	Interaction <interaction_name> failed. Manual recovery needed	Interaction is invalid	End of processing
515	Max number of auto-reply reached for interaction <interaction_name>	The maximum number of acknowledgements has been reached for the current interaction	End of processing
701	Unexpected error exception message	An unexpected error happened	Retry
710	Connection to database failed	Connection to database failed	Retry
711	Cannot get config of 'server'	E-Mail Server Java configuration error	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry
717	Config error: invalid option <option_name> in section <section_name>	Configuration error	Retry
718	Invalid custom field code key; <key name> is a reserved key	Invalid custom field code key	Retry
732	Invalid Tenant <tenant_name>	The identifier for the tenant is invalid	Retry
733	No tenant declared in CME for application <app_name>	The application has no tenant specified	Retry
734	Application <app_name> has more than one associated tenant (in CME)	This application has more than one tenant specified	Retry

6.x Compatible Operation Mode

[Figure 97](#) shows the E-mail Acknowledgement Receipt Properties dialog box with the 6.x Compatible Operation mode selected.



Figure 97: Acknowledgement Object, 6.x Compatible Operation Mode

When the Acknowledgement object in 6.x Compatible mode is placed in a strategy, you must specify a Standard Response. The actual Standard Response is stored in the UCS Database. URS cannot access the UCS Database, so a Script object for each receipt, CFGEmailAckReceipt, is stored in Configuration Server and points to its associated receipt in the UCS Database. The Standard Response library application synchronizes the Script objects in Configuration Server with the Acknowledgement receipt Standard Responses stored in the UCS Database. (In ICS, 6.5.x, the UCS Database is the conduit between Web Interaction Request Server and E-mail Server Java.)

Prior to completing the properties dialog box in 6.x Compatible mode, create the Standard Response Acknowledgement text in Response Manager, as described in the *Internet Contact Solution 6.5 Response Manager Help*.

Use the information in [Table 23](#) to complete the E-mail Acknowledgement Receipt Properties dialog box shown in [Figure 97](#).

Table 23: E-mail Acknowledgement Receipt, 6.x Mode

Parameter	Description
Acknowledgement Receipt	Select a receipt Standard Response from a drop-down list.
Include original message into reply	Select this check box to include the full text of the customer's message with the receipt sent to the customer.



[Figure 98](#) shows an event flow diagram for ICS 6.5.x when an Acknowledgement receipt is sent.

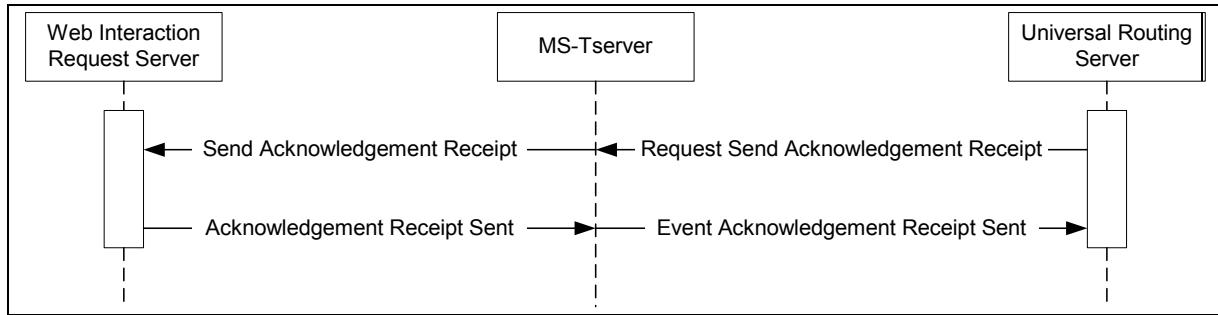


Figure 98: Event Flow Conceptual Diagram for Acknowledgement Receipt

URS and ICS Communication

Communication between URS and ICS components is accomplished using User Events issued on the virtual routing point to which the e-mail was submitted. Key-value pairs placed in T-Server User Events Extensions identify the event and include such things as protocol, event, reference ID, and others depending on the E-mail object. The value for the protocol key for both Acknowledgement receipts and automated responses is Autobot.

Important Information about URS/ICS Communication

- If URS sends a User Event to ICS components for an e-mail event previously submitted, E-mail Server Java disregards the duplicate events for that e-mail. This could occur if URS shuts down after submitting the event the first time but the e-mail is still pending when URS restarts.
- If your contact center expects a backlog of e-mails, the center can choose to have Acknowledgement receipts or automated responses sent immediately from E-mail Server Java's analysis process before the e-mail response is submitted to URS. A system administrator can set this ability in each Content Analysis rule. For automated responses to e-mails where intelligent e-mail classification is applied during content analysis, the administrator can also specify that responses with a value over the required relevancy are also automatically sent to a customer. See the ICS documentation for information about intelligent e-mail classification through the Standard Response library.

The following errors are possible for e-mail interactions:

- **-001 Timeout**—URS distributes the event but does not receive an answer from E-mail Server Java that it distributed the event within the 30-second timeout.
- **0001 Unknown error**—URS fails to distribute the event.

- 0013 Remote error—URS distributes the event and receives a negative answer error while requesting handling from E-mail Server Java. (The positive answer from E-mail Server Java would be that the request was handled normally.)

See the ICS documentation for information on ICS components and the configuration and installation of ICS software.



Attach Categories

This object enables you to manually emulate the results of the Classify object's request to Classification Server to attach one or more classification categories to an interaction (see [page 198](#)).

A *category* refers to a classification code in a Knowledge Manager Category tree used to categorize an interaction based on analysis of its content (see [Figure 99](#)).

The screenshot shows the Knowledge Manager-Content Analyzer interface. The title bar reads "Knowledge Manager-Content Analyzer: ConfigServer Host techpubs3; Port 6010; Application DavidsKM_7..". The menu bar includes File, Edit, View, Tools, and Help. The toolbar has icons for new, open, save, print, and help. The language is set to English. The main window has tabs for Resources and Category Tree. The Category Tree pane shows a hierarchy of categories under [Resources]: Auto Response, Bobs_Books (which contains books, cookery, fiction, periodicals, self-help, DSL Service, music), Bobs_Books_July, Chat Transcript, DegreesRUs, Financial service, Forwarding, and LanguageDetection. The cookery category is selected. The Subcategories of Category cookery pane shows a table with columns: Category Name, Owner, Date Modified, and Use in Classi. The Attached Standard Responses of Category cookery pane shows a table with columns: Standard R..., Status, Description, Acknowledg..., Autorespo..., Suggestion..., and a few rows of data. The Attached Rules of Category cookery pane shows a table with columns: Name, Rule Body, and Addresses To Me.

Figure 99: Knowledge Manager Categories and Standard Responses

Once classification categories are set up in Knowledge Manager, they carry over to Configuration Manager, where they appear in the under the Business Attributes > Category Structure folder (see [Figure 100](#)).

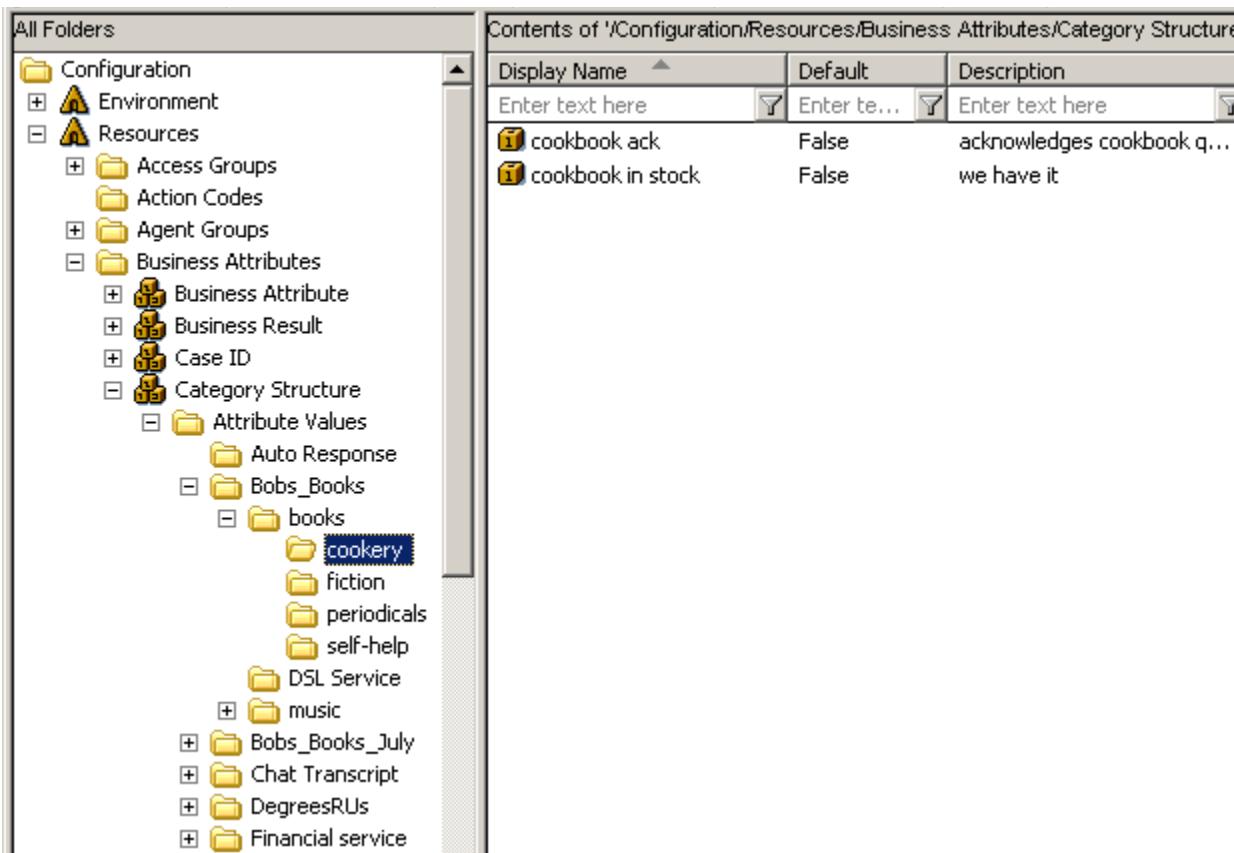


Figure 100: Configuration Manager Category Structure

Use Cases for Attach Categories

Note: The MultiScreen object (see [page 241](#)) allows you to attach categories that are associated with Standard Responses in Knowledge Manager.

Use the Attach Categories object (if you do not wish to use MultiScreen) when:

- Creating a strategy where screening results determine what categories to attach to an interaction with an Attach Categories object.
- Creating a strategy where an If object creates an expression based on categories manually attached to interaction with an Attach Categories object. URS analysis (true/false) of the expression determines the interaction path in the strategy.

You can also use categories to segment interactions (see “Classify” on [page 377](#)) or to select a Standard Response.

[Figure 101](#) shows an example completed Attach Categories Properties dialog box. It also shows the dialog box in which you select a category under the specified Root Name.

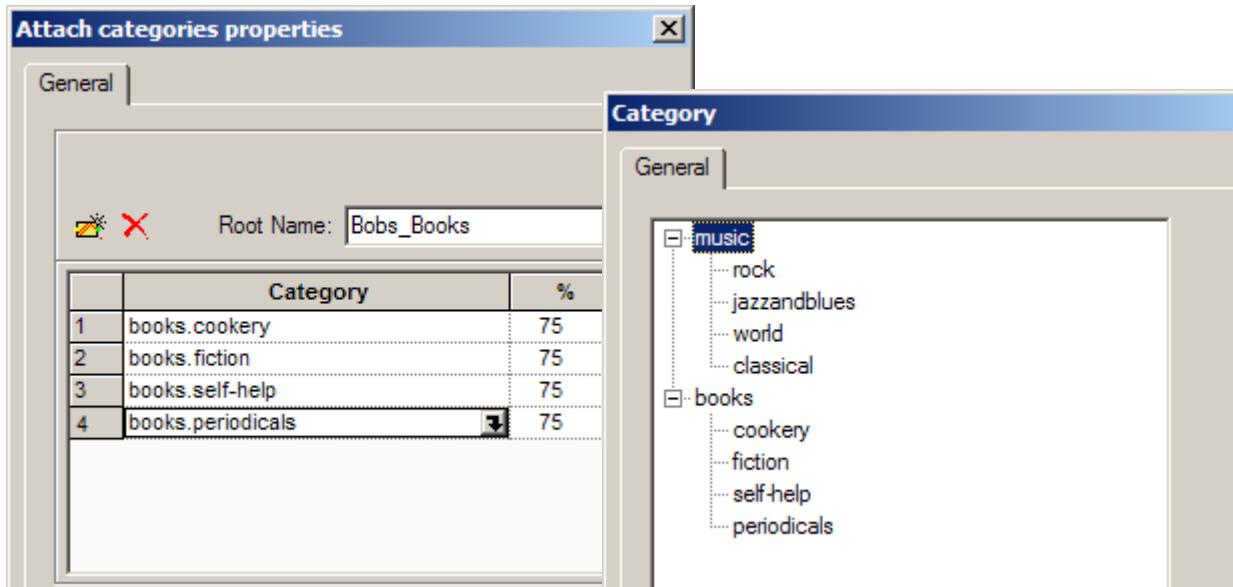


Figure 101: Attach Categories and Category Dialog Boxes

Use the information in [Table 24](#) to complete General tab in the Attach Categories Properties dialog box and the associated Category dialog box.

Table 24: Attach Categories Object, General Tab

Parameter	Description
Root Name	Select the Root folder containing the categories you wish to attach. The Root folder you select determines the categories that appear for selection in the Category dialog box shown in Figure 101 on page 180 . <ul style="list-style-type: none"> The Root Names that appear are initially configured in Knowledge Manager (see Figure 99 on page 178) and then carry over to Configuration Manager. You must specify a valid value or the strategy will not compile.
Category	Click the add item button to add a new entry (a small icon with a plus sign). After you click, a new row appears under Category. Click inside the row to bring up the Category dialog box shown in Figure 101 on page 180 . Available categories under the selected Root Name appear for selection. <ul style="list-style-type: none"> The categories that appear are initially configured in Knowledge Manager (see Figure 99 on page 178) and then carry over to Configuration Manager.

Table 24: Attach Categories Object, General Tab (Continued)

Parameter	Description
Category (continued)	Click a category to select it and then click OK. The Category dialog box closes and the category path (example: books.cookery) appears in the Attach Categories dialog box as a row under Category. Repeat these steps if you wish to attach another category to the interaction.
%	The default value for the percentage field is 75%. This field reflects the <i>minimum</i> relevancy the category must have in order for Classification Server to consider an interaction as belonging to that category. You have the option of changing the default but you cannot enter zero.



Autoresponse

Use the Autoresponse object to generate (but not send) an automatic response message using pre-written text from your Standard Response library.

The decision to send an automated response is affected by a number of conditions:

- The existence of an appropriate response—before an automated response is sent, it must have a high enough probability of being the correct answer.
- The customer segment—different customers may require different levels of service. You may not want to send an automated response to premier customers. If not, design the strategy to segment these customers along their own path in the strategy to avoid the Autoresponse object.
- The contact center hours of operation—you may want interactions received on off-hours (nights, weekends, and holidays) to generate an automated response while those received during working hours do not.
- Long queue time—when the waiting time for responding to interactions is long (whether due to a backlog of messages for agent or in Interaction Server), you may choose to send an automated response to a customer interaction.

Configuring the Autoresponse General Tab

The Autoresponse object has three Operation modes: Email, Open Media, and 6.x Compatible (see “6.x Compatible Operation Mode” on [page 176](#)).

E-Mail and Open Media Operation Modes

[Figure 102](#) shows the General tab of the Autoresponse Receipt Properties dialog box with the Open Media mode selected.

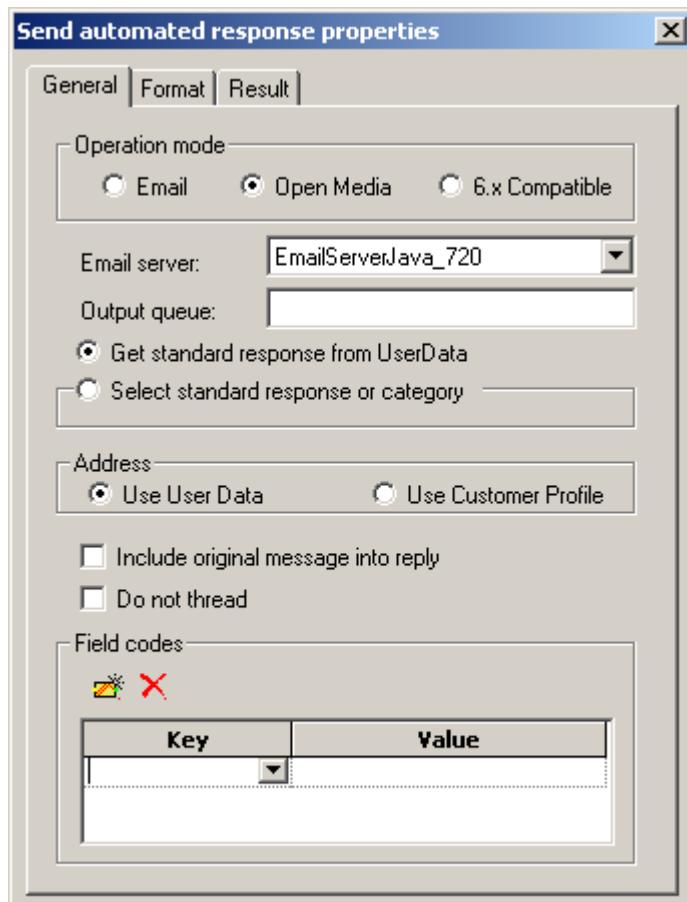


Figure 102: Autoresponse Object General Tab

The Email and Open Media modes use a tree viewer to represent pre-written Standard responses (see [Figure 103](#)).

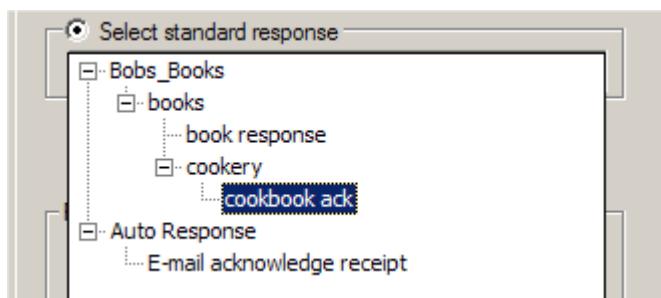


Figure 103: Tree Viewer for Autoresponse Standard Responses

Standard response names are carried over from Knowledge Manager. These names appear in Configuration Manager in the Business Attributes > Category Structure folder (see [Figure 104](#)).

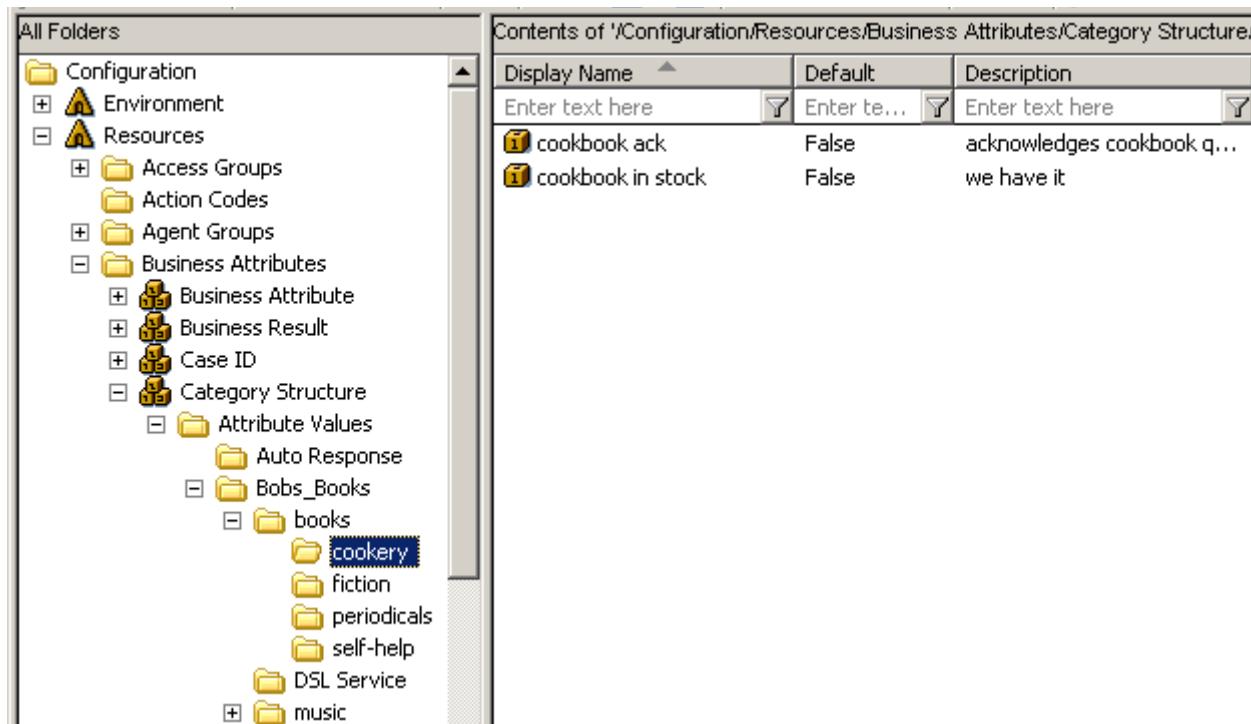


Figure 104: Category Structure Folder in Configuration Manager

Strategy-Linked Nodes from the Autoresponse Object

When a routing strategy in a business process uses the Autoresponse object, the specified E-mail Server Java and interaction output queue (see Figure 88 on [page 162](#)) for the autoresponse e-mail appear as strategy-linked nodes (as shown in the Auto Response and Outbound e-mails objects in [Figure 105](#)).

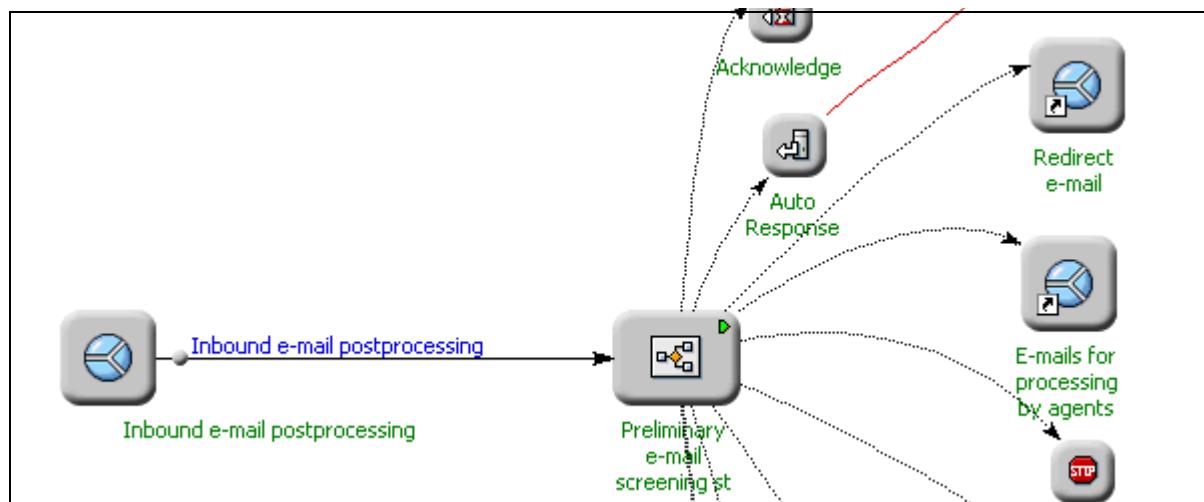


Figure 105: E-mail Server Java and Output Queue in a Business Process

Use the information in [Table 25](#) to complete the fields in the General tab of the Send automated response properties dialog box in Email or Open Media mode.

Table 25: Autoresponse Properties for E-Mail and Open Media Modes

Parameter	Description
Operation mode E-Mail	Select if this autoresponse will be sent by a standard instance of E-Mail Server Java.
Operation mode Open Media	Select if this autoresponse will be sent by an E-Mail Server Java modified to handle Open Media interactions.
E-mail Server	Click the down arrow and select the Application name in Configuration Manager for the E-mail Server Java to be notified. <ul style="list-style-type: none"> • You can leave this field empty. If empty, Interaction Server uses the first available E-mail Server Java named in its Connections list.
Output queue	Required. Click in the Output queue text box to expand the business processes. Select the interaction queue where E-mail Server Java should place the autoresponse e-mail.
Get standard response from UserData	If you select this radio button, the Select Standard Response radio button is disabled and vice-versa. Uses the classification codes returned by the Classify object and stored in the interaction User Data to get a Standard Response. <ul style="list-style-type: none"> • To use this field, the following key must be present in the interaction User Data: CategoryId. If SRLid and CtgId in the Request message and CategoryId in UserData are null, then an error is generated and the interaction goes out the Error port. <p>Note: If you select this option instead of the alternative, Select Standard Response from User Data, you cannot use the field codes functionality described below.</p>
Select standard response or category	Use this option to manually select a Standard Response (such as cookery in Figure 90 on page 163) or category (such as Bobs_Books or books in Figure 90 on page 163) to indicate the pre-written text to be included in the autoresponse to the customer. Standard responses are initially defined in Knowledge Manager, stored in the UCS Database, then carried over to Configuration Manager. For more information on Standard Responses, see the <i>eServices (Multimedia) 8.1 User's Guide</i> . <p>Notes:</p> <ul style="list-style-type: none"> • If you select this option, you can use the field codes functionality described below (if the selected Standard Response contains one or more field code custom variables).

Table 25: Autoresponse Properties for E-Mail and Open Media Modes (Continued)

Parameter	Description
Address (Open Media mode only)	<p>Select whether to take the address to which the autoresponse should be sent from user data attached to the interaction or from the customer's profile stored in the UCS Database.</p> <p>If you choose to take the address from the database, you must place the Identify Contact object in the strategy before the Autoresponse object. For information on the Identify Contact object, see "Identify Contact" on page 237.</p> <p>Note: This pane does not appear in E-Mail mode.</p>
Include original message into reply	Select the check box if the autoresponse should attach the original message.
Do not thread	If this box is checked, it creates a new entry in the UCS Database that is not part of the original thread.
Field codes	<p>You can use this pane to assign values to any field code custom variables that may have been associated with the Standard Response in Knowledge Manager. See the <i>eServices (Multimedia) 8.1 User's Guide</i> for more information on field codes.</p> <p>To assign field code values, click the new item button (). A new row appears under the Key and Value headings. Under Key, existing field code custom variables associated with the response appear for selection. Under Value, any existing strategy variables display for selection. Use the information in the "Field Codes and Custom Variables" section below to edit the values.</p> <p>Note: You can also manually enter field codes and values.</p>

Autoresponse Format Tab

You can customize the To, From, and Subject areas in the Standard Response message to the customer. Figure 94 on [page 169](#) shows the Format tab for the Acknowledgement object. The Format tab appears the same for the Autoresponse object. Use the information in Table 18 on [page 170](#) to complete the fields in the Format tab.

Autoresponse Result Tab

[Figure 106](#) shows the Result tab in the Send Automated Response Properties dialog box when Assign values of the key-value pair is checked.

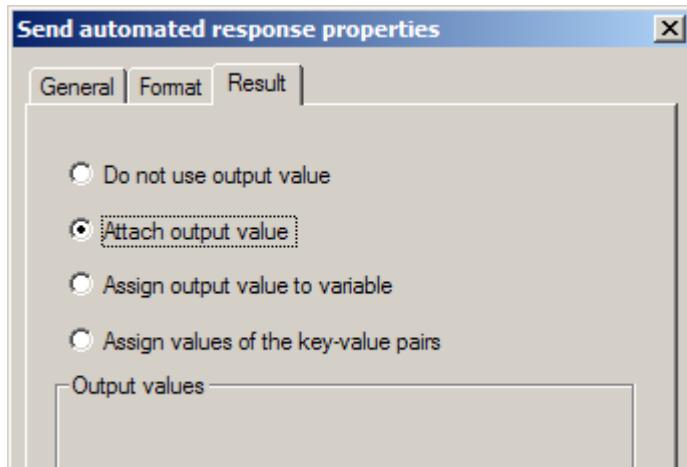


Figure 106: Autoresponse Object Result Tab

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Use the information in Table 20 on [page 173](#) to complete the Result tab in the Send Automated Response Properties dialog box.

Returned Results for the Autoresponse Object

The results returned will be either an Event3rdServerResponse message or an Event3rdServerFault message (see Table 27 on [page 187](#))

The content of the Event3rdServerResponse message is shown in [Table 26](#):

Table 26: Autoresponse Object Returned Results

Key	Description
NewInteractionId	The interaction ID of the new outbound e-mail.
SrlId	Global unique Standard Response identifier if it was found and used.

Any fault message returned uses the fault codes shown in [Table 27](#):

Table 27: Autoresponse Object Fault Codes

FaultCode value	FaultString value	Description	Action
105	No text found in Standard Response <SR_name>	The Standard Response exists but it contains no text	Exit from strategy
106	No ‘From/To’ address found	Either the From or the To address is missing	Exit from strategy
107	Too many ‘From/To’ addresses found	Either the From or the To field (or both) contain too many addresses	Exit from strategy
108	‘incorrect name’. Invalid ‘address’	The address or the domain is invalid	Exit from strategy
109	Standard response <SR_name> validity period ‘not yet started/expired’	The period during which the specified Standard Response is valid has not started or has ended	Exit from strategy
111	Standard response <SR_name> not found	This Standard Response does not exist in the database.	Exit from strategy
112	No Standard Response found in database	The database has no Standard Responses in it	Exit from strategy
201	Missing parameter name	One parameter is missing	Exit from strategy
202	Parameter ‘1’ and ‘2’ are not allowed	These two parameters cannot be present simultaneously	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
204	Incorrect value for parameter <parameter_name>, expected value 1 but was value 2	The specified parameter has an invalid value	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected message from the server’s client	End of processing

Table 27: Autoresponse Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
510	Object not found in database	Object is not found in database	End of processing
513	Interaction <interaction_name> failed. Manual recovery needed	Interaction is invalid	End of processing
515	Max number of auto-reply reached for interaction <interaction_name>	The maximum number of autoresponses has been reached for the current interaction	End of processing
701	Unexpected error exception message	An unexpected error happened	Retry
710	Connection to database failed	Connection to database failed	Retry
711	Cannot get config of 'server'	E-Mail Server Java configuration error	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry
717	Config error: invalid option <option_name> in section <section_name>	Configuration error	Retry
718	Invalid custom field code key; <key name> is a reserved key	Invalid custom field code key	Retry
732	Invalid Tenant <tenant_name>	The identifier for the tenant is invalid	Retry
733	No tenant declared in CME for application <app_name>	The application has no tenant specified	Retry
734	Application <app_name> has more than one associated tenant (in CME)	This application has more than one tenant specified	Retry

6.x Compatible Operation Mode

Figure 107 shows the properties dialog box for the Autoresponse object with the 6.x Compatible mode selected.

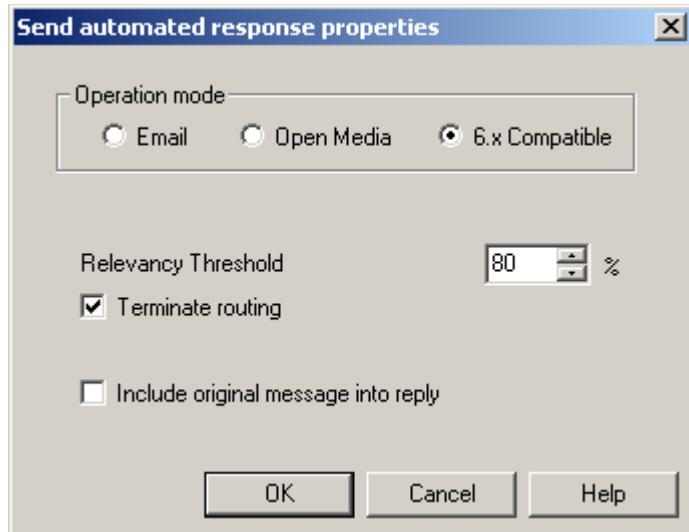


Figure 107: Autoresponse 6.x Compatible Operation Mode

Note: Also see “URS and ICS Communication” on [page 177](#).

When a customer sends an interaction, E-mail Server Java analyzes it and calculates a relevancy value—the percentage of certainty that a Standard Response is the correct answer to the customer’s inquiry.

Note: You can expand the analysis capabilities of E-mail Server Java by obtaining Content Analyzer and Classification Server, which work with E-mail Server Java. Content Analyzer enables more detailed content analysis and can learn from previous agent’s selections to provide greater accuracy in the responses suggested.

Once an automated response is selected, a value associated with this response is attached to the interaction as the `ICC_MaxSuggestedResponseRelevancy` attached data key. URS reads the value and compares this to the `Relevancy Threshold` set by the user in the Autoresponse object. If the relevancy value is equal to or higher than the `Relevancy Threshold`, URS requests E-mail Server Java to send the response. If the value is below the `Relevancy Threshold`, URS does not request that E-mail Server Java send the response. In this circumstance, the strategy must contain another Routing object so that the interaction can be routed to an agent when no response is sent.

Completing the Properties Dialog Box

Prior to selecting 6.x Compatible mode in the Send Automated Response Properties dialog box shown in [Figure 107](#), set up automated response text using the Standard Response library application that is a part of Response Manager. See ICS documentation for more information.

Use the information in [Table 28](#) to complete the Send Automated Response Properties dialog box: in 6.x mode:

Table 28: Send Automated Response, 6.x Mode

Parameter	Description
Relevancy Threshold	Click the arrows in front of the percentage sign to specify the minimum threshold that the relevancy value must have for URS to send a request for an automated response. E-mail Server Java attaches the relevancy value to the interaction after performing a content analysis.
Terminate Routing	Select this option if interactions should not continue being routed after an automated response is sent. When this option is selected (default), URS sends a RequestRouteCall message with Type=Rejected after the automated response is sent. When the Terminate Routing is cleared, routing continues through the strategy, enabling other routing rules to be applied for routing the interaction to an agent.
Include original message into reply	Select this check box to include the full text of the customer's message with the automated response.

Example of Using Autoresponse in 6.x Compatible Mode

The Relevancy Threshold is set to 80 percent in the Autoresponse object. A customer e-mail is received by E-mail Server Java. E-mail Server Java analyzes the e-mail and discovers an automated response that has a relevancy value of 85 percent. E-mail Server attaches this value to the e-mail. When URS encounters this Autoresponse object in the strategy, URS compares the Relevancy Threshold to the relevancy value attached to the e-mail and determines that the relevancy value is higher than the Relevancy Threshold. URS requests that E-mail Server send the automated response.

[Figure 108](#) shows an ICS 6.5.x event flow conceptual diagram for an automated response that meets relevancy value requirements.

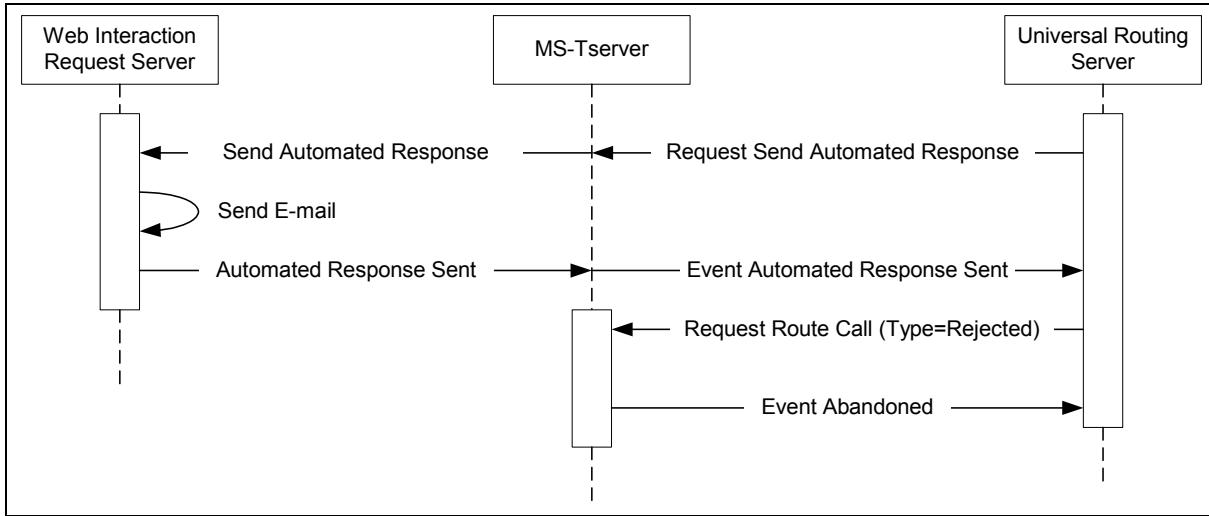


Figure 108: Event Flow Conceptual Diagram for Automated Response

Note: If an agent response is required, instead of URS sending RequestRouteCall with Type = Rejected and MS T-Server responding with EventAbandoned, URS sends RequestRouteCall with the target equal to an agent.

If URS does not receive any event from MS T-Server within a timeout of 30 seconds, the e-mail interaction is routed to an agent when additional routing rules are specified, as if there were no automated response to send.



Chat Transcript

Use the Chat Transcript object to generate (but not send) an e-mail with text taken from your Standard Response library, with the customer's chat transcript attached. This object causes URS to generate a request to E-mail Server Java for the method `ChatTranscript`.

Use Cases

The customer fills out a form on the corporate web site, including filling in a field to indicate the chat alias and the client network (in other words, `Username: MyFineFriend Client: AOL`).

1. The form is routed to an agent via URS executing a routing strategy (as usual for chat).
2. The agent gets a preview screen (“screen pop”).
3. The agent contacts the customer using the chat application within the Genesys Desktop.

4. Customer accepts the chat session.
5. At the end of the session, the customer requests a transcript of the session, which can be sent via e-mail.
6. The agent conducting the chat session requests that the chat transcript be sent to the customer.

Chat Transcript General Tab

Figure 109 shows the General tab of the dialog box that opens when you click the Chat Transcript button and place the object in your strategy.

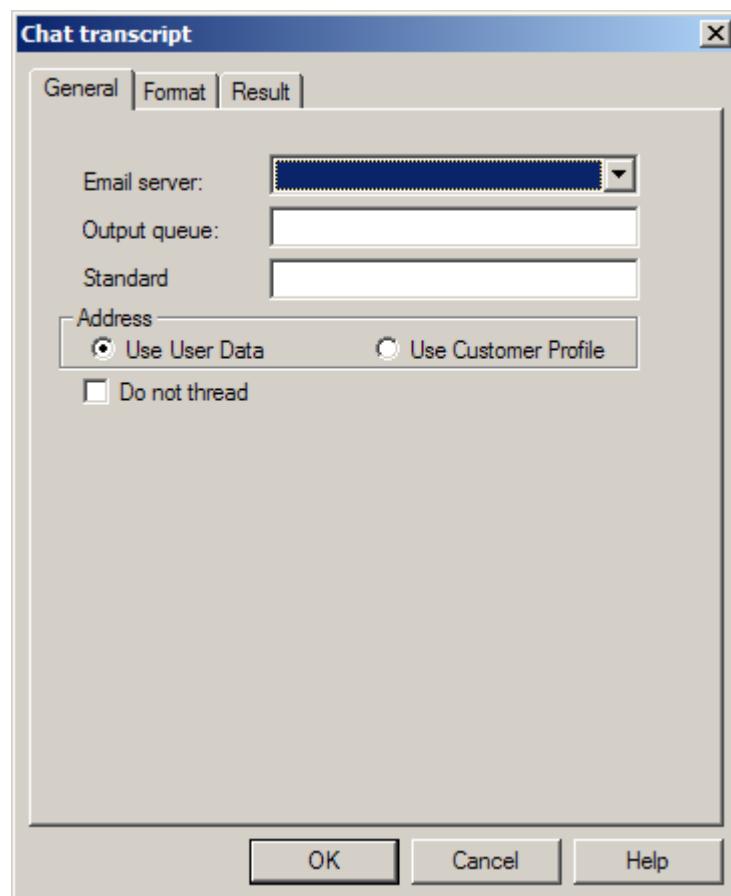


Figure 109: Chat Transcript Dialog Box, General Tab

Use the information in [Table 29](#) to complete the General tab in the Chat Transcript Properties dialog box.

Table 29: Chat Transcript Object, General Tab

Parameter	Description
E-mail Server	Click the down arrow and select the Application name in Configuration Manager for the E-mail Server Java that URS should notify (via Interaction Server) with an Event3rdServerResponse message. This field may be left empty. If empty, Interaction Server uses the first available E-mail Server Java named in its Connections list.
Output Queue	Required. Click in the Output queue text box to expand the business processes. Select the interaction queue where E-mail Server Java should place the e-mail with the attached chat transcript.
Standard	Use this option to manually select an identifier for an AutoResponse Business Attribute representing pre-written text to be used as an automated reply (see Figure 90 on page 163). Standard responses are initially defined in Knowledge Manager, stored in the Universal Contact Server database, then carry over to Configuration Manager.
Address Use UserData	Select this button to have E-mail Server Java use the interaction User Data and the _OutboundToAddress key for the customer's e-mail address. See " Special Note on Use UserData Field " below.
Use Customer Profile	Select this button to have E-mail Server Java use the customer's profile in the UCS Database to get the customer's e-mail address.
Do not thread	If this box is checked, it creates a new entry in the UCS Database that is not part of the original thread.

Special Note on Use UserData Field

To be able to send a chat transcript, E-mail Server Java needs a “send to” e-mail address. This e-mail address may be obtained in one of the following ways:

1. A web site can be custom-developed to automatically include the customer's e-mail address (it is mandatory to specify it when a client is requesting a chat session).
2. In a routing strategy before using the Chat Transcript object, the e-mail address can be pulled out through regular IRD/URS methods and attached as UserData to an e-mail interaction.
3. An agent can manually perform the Attach UserData action on the desktop with the _OutboundToAddress key.

Warning! If none of these actions are performed and instead you use the Genesys eServices default settings, the `_OutboundToAddress` key will be missing in the interaction event. The Chat Transcript object cannot be used with `UseAddressFromUserData` set to true because, with this option, the customer's e-mail address is expected to be under the `_OutboundToAddress` key. If this key is missing, an error message is returned by E-mail Server Java and the chat transcript will not be e-mailed to the customer.

Chat Transcript Format Tab

You can customize the `To`, `From`, and `Subject` areas in the response e-mail to the customer. [Figure 110](#) shows the Format tab for Chat Transcript.

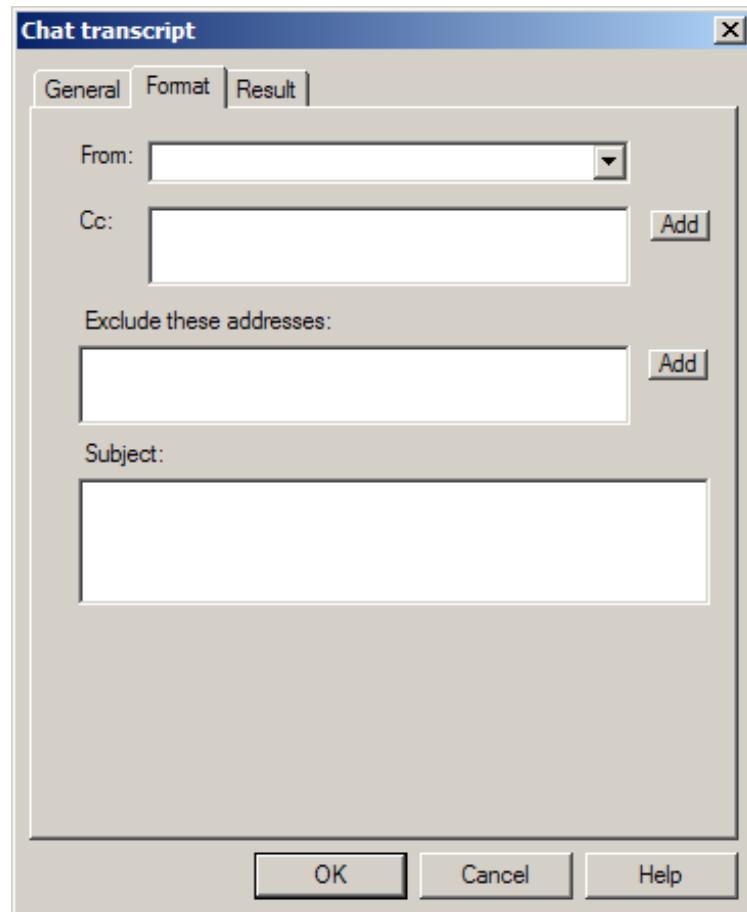


Figure 110: Chat Transcript Object, Format Tab

Use the information in [Table 30](#) to complete the Format tab.

Table 30: Chat Transcript Format Tab

Parameter	Description
From	If applicable, click the From: down arrow and select a From address for the Standard Response. Any listed names are E-mail Accounts Business Attribute.
Cc	If applicable, click Add opposite Cc and select an address to copy in on the Standard Response. Any listed names are E-mail Accounts Business Attribute.
Exclude these addresses	The third area lists addresses found in the original e-mail. If applicable, click Add and select addresses to exclude in the Standard Response. Click Variable if the addresses are contained in a variable. Can contain a comma-separated list of domains, aliases, variables, and e-mail addresses.
Subject	If applicable, enter a new Subject for the Standard Response.

Chat Transcript Result Tab

[Figure 111](#) shows the Result tab of the dialog box that opens when you click the Chat Transcript button and place the object in your strategy.

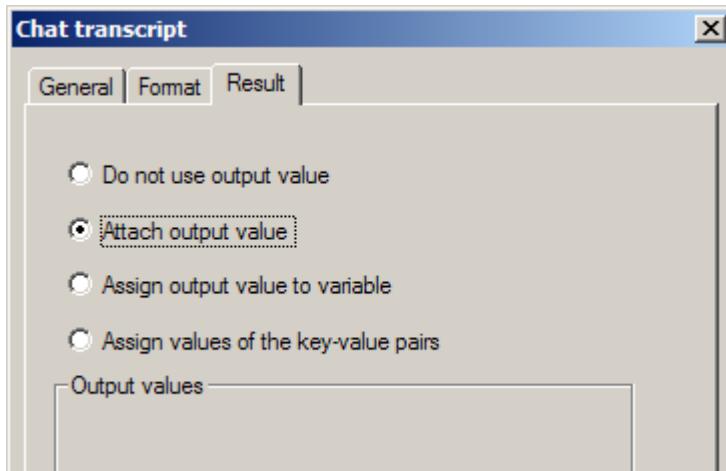


Figure 111: Chat Transcript Dialog Box, Result Tab

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Use the information in Table 20 on [page 173](#) to complete the Result tab in the Chat Transcript Properties dialog box

Returned Results for the Chat Transcript Object

The results returned will be either Event3rdServerResponse (see [page 151](#)) or a fault message (Event3rdServerFault, see [page 151](#)).

The content of the Event3rdServerResponse message is shown in [Table 31](#):

Table 31: Chat Transcript Object Returned Results

Key	Description
SrlId	Global unique Standard Response identifier if it was found and used.
Address	The e-mail address used to send the transcript.

Any fault message returned uses the fault codes shown in [Table 32](#):

Table 32: Chat Transcript Object Fault Codes

FaultCode value	FaultString value	Description	Action
106	No ‘From/To’ address found	Either the From or the To address is missing	Exit from strategy
107	Too many ‘From/To’ addresses found	Either the From or the To field (or both) contain too many addresses	Exit from strategy
108	‘incorrect name’. Invalid ‘address’	The address or the domain is invalid	Exit from strategy
109	Standard response <SR_name> validity period ‘not yet started/expired’	The period during which the specified Standard Response is valid has not started or has ended	Exit from strategy

Table 32: Chat Transcript Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
201	Missing parameter name	One parameter is missing	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected message from the server's client	End of processing
510	Object not found in database	Object is not found in database	End of processing
701	Unexpected error exception message	An unexpected error happened	Retry
710	Connection to database failed	Connection to database failed	Retry
711	Cannot get config of 'server'	E-Mail Server Java configuration error	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry
717	Config error: invalid option <option_name> in section <section_name>	Configuration error	Retry
718	Invalid custom field code key; <key_name> is a reserved key	Invalid custom field code key	Retry
732	Invalid Tenant <tenant_name>	The identifier for the tenant is invalid	Retry
733	No tenant declared in CME for application <app_name>	The application has no tenant specified	Retry
734	Application <app_name> has more than one associated tenant (in CME)	This application has more than one tenant specified	Retry



Classify

Use the Classify object to request Classification Server to assign one or more classification categories to an interaction.

A *category* is a classification code in a category tree, originally set up in Knowledge Manager (see Figure 99 on [page 178](#)) for the purpose of Classification Server categorizing an interaction based on analysis of its content. You can then segment interactions based on the resulting categories (see “Classify” on [page 377](#)) or use them to select a Standard Response. You can also manually assign classification categories using the Attach Categories object (see [page 178](#)).

Note: If your site’s categories are not yet defined in Knowledge Manager (see Figure 99 on [page 178](#)), you may wish to use the Screen object, which performs word matching (see [page 263](#)) instead of the Classify object.

Once the classification request is made:

- If Classification Server finds a category with the minimum specified relevancy (see Confidence in Table 33 on [page 200](#)), the results are processed based on the option you select in the Result tab (see Figure 116 on [page 205](#)) and the interaction goes through the green port. See [page 201](#) for more information on returned results.
- If an existing category cannot be found based on analyzing the interaction, the null result is attached to the interaction and the interaction goes through the green port (the error is not retrievable through the current Error object). See “Event3rdServerFault” on [page 203](#) for more information.

Classify General Tab

Figure 112 on [page 199](#) shows the General tab.

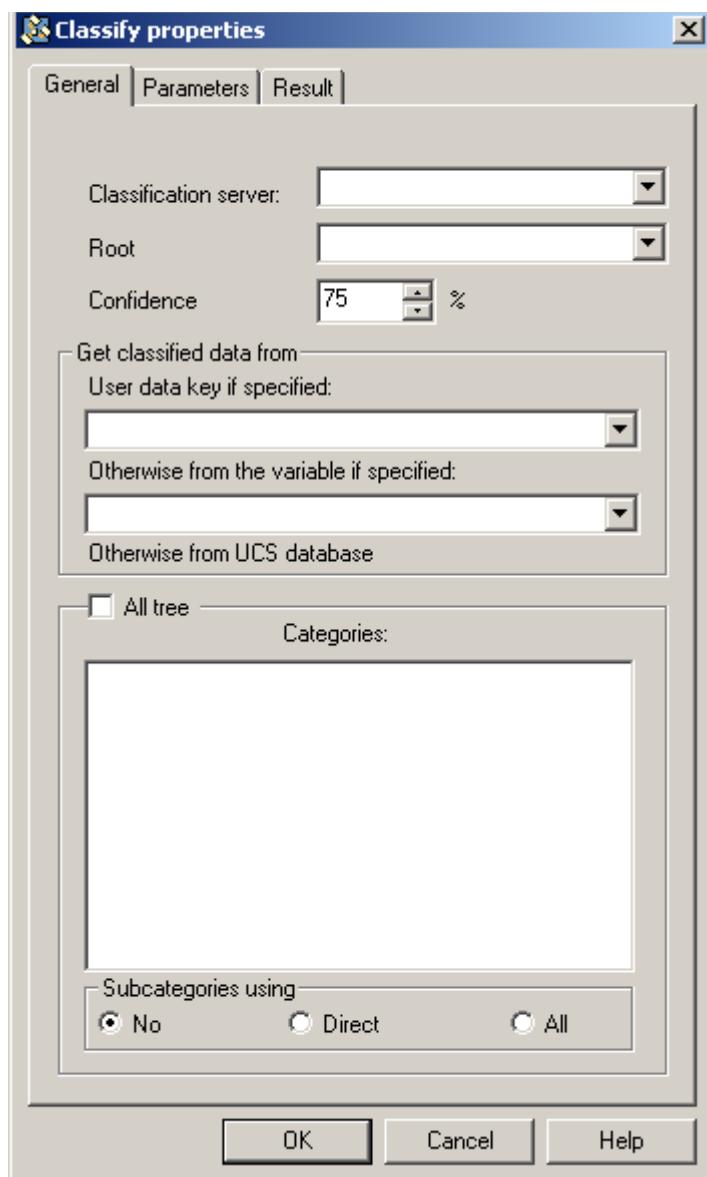


Figure 112: Classify Object, General Tab

Use the information in [Table 33](#) to complete the `General` tab in the `Classify` dialog box.

Table 33: Classify Object, General Tab

Parameter	Description
Classification	<p>Select the Application name of the Classification Server in the Configuration Server database.</p> <ul style="list-style-type: none"> This field may be left empty. If empty, Interaction Server uses the first available Classification Server named in its Connections list.
Root	<p>Required field. Specify the overall classification category by selecting a top level folder in Configuration Manager under Resources > Category Structure > Attribute Values (see Figure 113 on page 201). Must contain a valid value, or the strategy will not compile.</p>
Confidence	<p>Enter a number from 1-100 reflecting the minimum relevancy each classification category must have in order for Classification Server to consider an interaction as belong to that category. The default value is 75%. You cannot enter a zero value.</p>
Get classified data from	<p>This pane enables you to select the source of the classification data that the Classify object will use to direct interactions. You can retrieve this data from user data attached to the interaction, from a variable attached to the interaction, or from the UCS Database. If both text boxes are left empty, IRD looks for the classification data in the UCS Database.</p>
All Tree	<p>The categories tree contains the categories you created in Knowledge Manager, read from the UCS Database.</p> <ul style="list-style-type: none"> If you select All Tree, the Categories tree (see Figure 112 on page 199) is disabled. Classification Server looks for all categories under the specified Root.
Categories	<p>If All Tree is cleared, the Categories tree is enabled so you can select individual subcategories. Select at least one subcategory under the Root.</p>
Subcategories using	<p>The Subcategories using pane determines whether Classification Server should consider parent and child Categories.</p> <p>The Subcategories using pane is enabled if All Tree is cleared. These options control how Classification Server uses the category tree.</p> <ul style="list-style-type: none"> Select the No radio button to have Classification Server consider only the categories you selected in the tree and not to include any child categories. Select the Direct radio button to have Classification Server consider only the direct children of selected (parent) categories. Select the All radio button to have Classification Server consider all children of selected (parent) categories.

Category Structure

[Figure 113](#) shows an example Category Structure tree and Root folder in Configuration Manager. In this example, InternetDealership is the Root category.

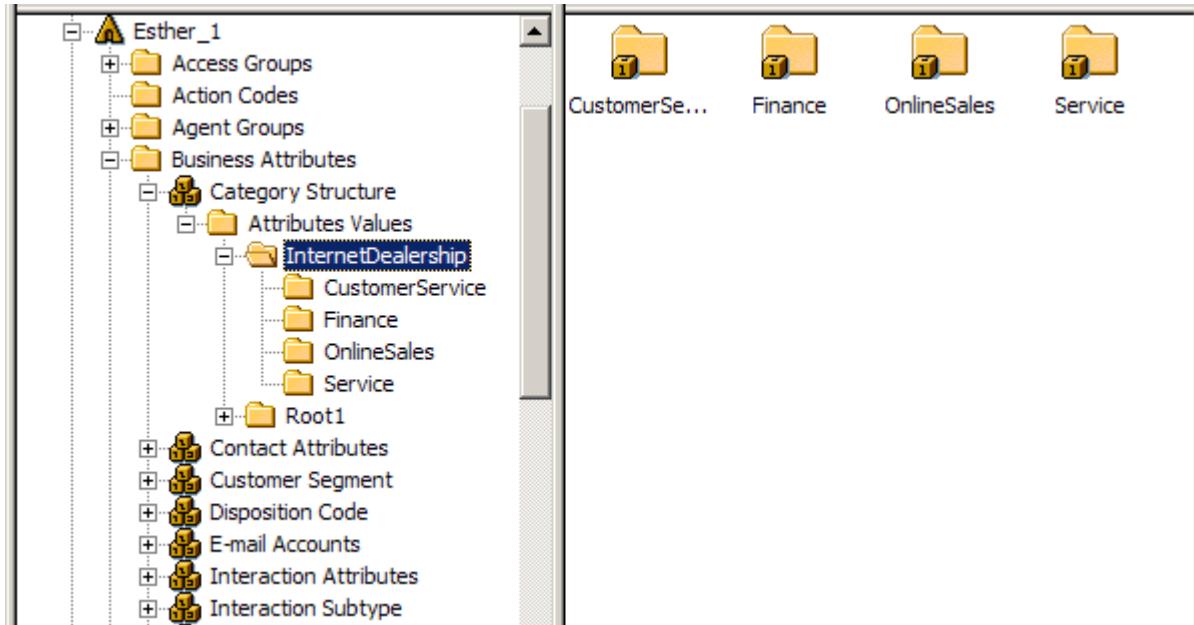


Figure 113: Attribute Values Folder, Example Categories

- In this simplified example there is only one Root category, [InternetDealership](#), but in many cases, there will be more than one Root.
- The Categories pane in the [Classify](#) dialog box shows all the categories in Configuration Manager under the Root you selected. The categories under the [InternetDealership](#) Root in Figure 113 on [page 201](#) are [CustomerService](#), [Finance](#), [OnlineSales](#), and [Service](#).

Returned Results for the Classify Object

[Figure 114](#) shows a diagram of results returned from Classification Server including the message types.

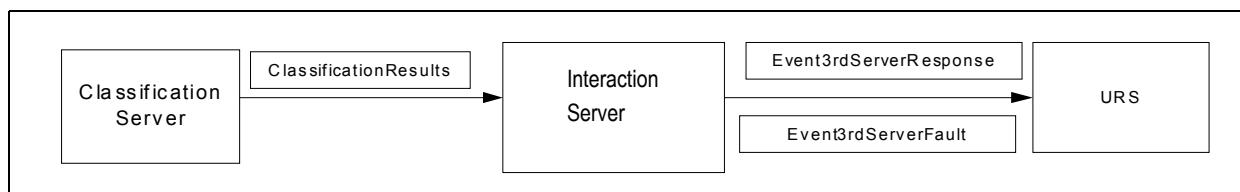


Figure 114: Results Returned with Classify Object

Event3rdServerResponse

The results returned from Classification Server in Event3rdServerResponse will have:

CtgId	An ID of a Category with a highest relevancy.
CtgRelevancy	The relevancy of a Category presented in CtgId.
CtgName	Name of the Category with the highest relevancy.
Array of classification pairs	
	A key-value pair that has CtgId as a key and corresponding relevancy percentage as a value.

The standard way to deal with the classification request results is to attach them to the interaction's User Data in the Result tab (see Figure 116 on [page 205](#)).

Attaching Results to User Data

While you can assign Classify object results to a variable, Genesys does not recommend this. The recommended way of dealing with the classification results is to attach them to the interaction. Then User Data will have the keys listed in the table below with the corresponding values returned by Classification Server. As an example, User Data would have the following pairs after the attachment (see [Table 34](#)):

Table 34: User Data Example

Parameter	Value
CtgId	00001a05F5U900QW
CtgRelevancy	95
CtgName	Cooking
CtgId_00001a05F5U900QW	95
CtgId_00001a05F5U900QX	85
CtgId_00001a05F5U900QY	75
CtgId_00001a05F5U900QZ	65

See [Supplement](#) for more details. In the event that an existing category cannot be found based on analyzing the interaction, the Event3rdPartyReturn value indicates no match and the interaction proceeds through the green port.

Note: A routing strategy must also direct interactions of unknown categories to a target, such as to an interaction queue read by a supervisor. For this reason, you may want to define an “unknown” or “non-deliverable” category in Knowledge Manager (see Figure 99 on [page 178](#)).

Event3rdServerFault

If the message cannot be transmitted correctly, the resulting Event3rdServerFault message will use the fault codes described in [Table 35](#).

Table 35: Classify Object Fault Codes

FaultCode value	FaultString value	Description
501	Internal error in application.	General internal error happened in third party application during request processing. For more details see the application log.
502	Unexpected third party protocol message type=<value>	Protocol parser encountered unexpected protocol message from server’s client.
504	Cannot find interaction= <request message Id>	Server cannot find specified interaction.
505	Classification functionality is not licensed	Server cannot find license for classification functionality. All classification requests will be aborted.
506	There is no default model for: (<Tenant>, <Language>, <Category Root>)	Server can’t find default (active) model for specified (<Tenant>, <Language>, <Category Root>).
511	ucsapi exception: <description>, interaction= <request message Id>	UCS API function exception.

Classify Parameters Tab

[Figure 115](#) shows the Parameters tab of the Classify Properties dialog box.

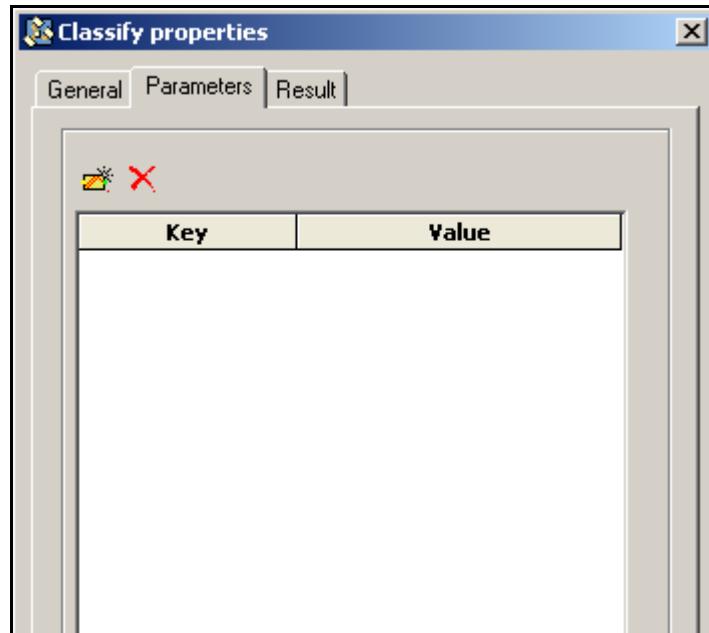


Figure 115: Classify Properties Dialog Box, Parameters Tab

Reserved for the future releases of eServices.

Classify Result Tab

[Figure 116](#) shows the Result tab for the Classify properties dialog box.

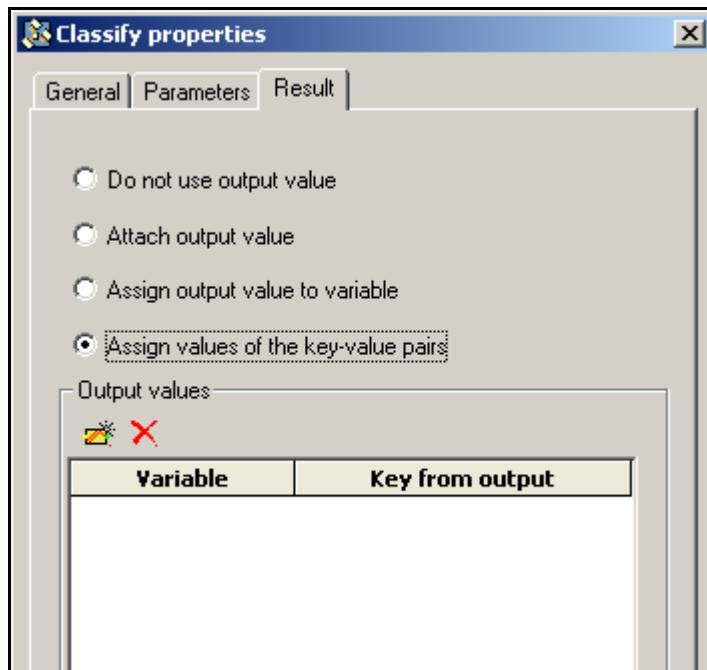


Figure 116: Classify Object, Result Tabs

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Use the information in Table 20 on [page 173](#) to complete the Result tab in the Send Automated Response Properties dialog box



Create Interaction

Use the Create Interaction object to request Universal Contact Server (UCS) to create an interaction in the UCS Database and associate it with an existing customer contact or with a newly created contact. You can also specify the content of the interaction. The UCS Database can store three types of interaction content simultaneously:

- Text. Interactions usually consist of plain text that does not need special tools or formatting in order to display to the agent.
- Structured Text, such as .html or .xml files.

- Binary Data. Any binary data associated with an interaction, such as an e-mail attachment, fax, or image.

Create Interaction General Tab

Figure 117 shows the properties dialog box for the Create Interaction object already filled in with sample data.

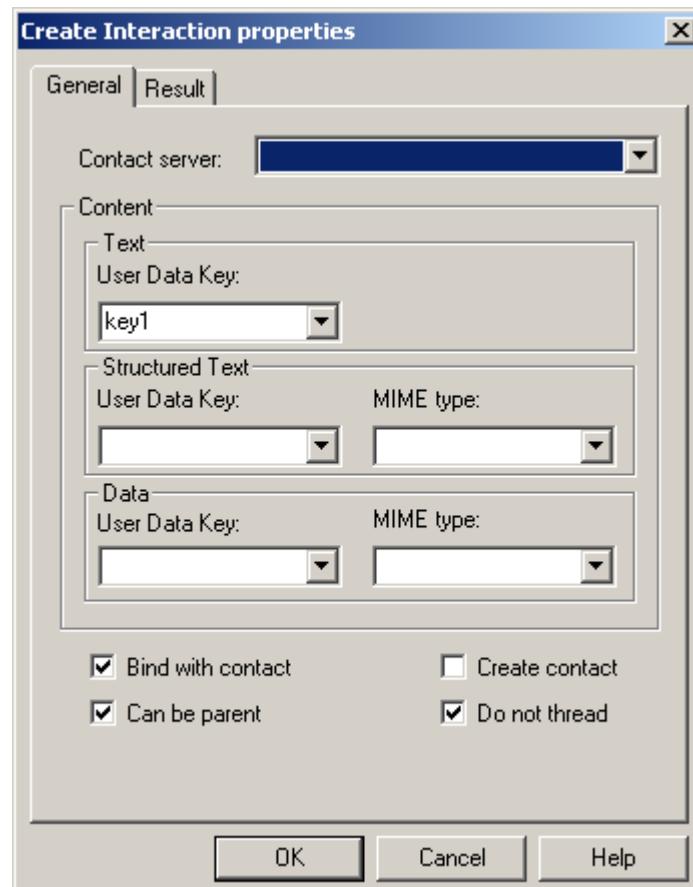


Figure 117: Create Interaction Properties Dialog Box

Use the information in [Table 36](#) to complete the General tab of the properties dialog box.

Table 36: Create Interaction, General Tab

Parameter	Description
Contact Server	Click the down arrow and select a UCS Application from those configured in Configuration Manager.
Content Text, User Data Key	Click the down arrow and select a variable that contains the User Data key or enter a string for the User Data key contained in the incoming interaction (see “User Data” on page 152). The value of User Data that corresponds to key specified in this field will be stored in UCS Database as an interaction’s Plain Text.
Content Structured Text, User Data Key	Click the down arrow and select a variable that contains the User Data key or enter a String for the User Data key contained in the incoming interaction (see “User Data” on page 152). The value of User Data that corresponds to the key specified in this field will be stored in the UCS Database as interaction’s Structured Text.
Content, Structured Text, MIME Type	<ul style="list-style-type: none"> Note: MIME refers to Multipurpose Internet Mail Extensions, which is a common method for transmitting non-text files via Internet e-mail. <p>Enter the MIME type or select from the list to indicate a type of interaction content under the category of Structured Text. Select one of the following: text/html, text/plain, text/richtext, text/html.</p> <p>For more complete list of MIME types see: http://www.iana.org/assignments/media-types/</p> <p>What you select here assists the Agent Application in choosing the proper application to process or display this part of interaction. This field cannot be empty if Structured Text, User Data Key contains a value.</p>
Content, Data, User Data Key	Note: <i>Data</i> refers to binary data. Click the down arrow and select a variable that contains the User Data key or enter a String for the User Data key contained in the incoming interaction (see “User Data” on page 152). The value of User Data that corresponds to the key specified in this field will be stored in the UCS Database as an interaction’s binary content.

Table 36: Create Interaction, General Tab (Continued)

Parameter	Description
Content, Data, MIME Type	<p>For a complete list of MIME types, see the above link. Enter the MIME type or select a value to indicate the interaction content type under the category of Data:</p> <p>application/msword, application/octet-stream, application/postscript, application/rtf, application/vnd.ms-powerpoint, application/vnd.ms-project, application/vnd.visio, application/voicexml+xml, application/xml, application/xml-dtd, application/zip, audio/basic, audio/mpeg, audio/mpeg4-generic, image/g3-fax, image/gif, image/jpeg, image/tiff, message/delivery-status, message/http, message/news, message/partial, message/rfc822, message/sip, message/sipfrag, message/tracking-status, multipart/alternative, multipart/form-data, multipart/mixed, multipart/parallel, multipart/voice-message, text/html, text/plain, text/richtext, text/xml, video/DV, video/JPEG, video/MPEG, video/mpeg4-generic, video/quicktime, video/raw.</p> <p>This information helps Agent Application to choose the proper application to process or display this part of the interaction. This field cannot be empty if Data, User Data Key contains a value.</p>
Bind with contact	<p>If selected, UCS will look for a customer contact record and associate it with the interaction. If a contact record is not found, it will be created. By default this option is cleared.</p>
Create contact	<p>If selected, instructs to create a new contact record in the UCS Database.</p> <ul style="list-style-type: none"> • If Bind with contact is not checked, Create contact is disabled. • if Bind with contact is checked, Create contact is enabled and checked by default. <p>By default this option is checked, but grayed out.</p> <p>An unchecked value of the Create Contact check box in the Create Interaction object is not supported (ignored) by UCS.</p>
Can be parent	<p>If selected, indicates this interaction can have child interactions. By default, this option is selected.</p>
Do not thread	<p>If selected, instructs to not thread under another interaction (which is specified by under key ParentID in UserData) in the contact's history in the UCS Database. By default this option is selected.</p>

Create Interaction Result Tab

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Use the information in Table 20 on [page 173](#) to complete the Result tab in the Create Interaction Properties dialog box

Creating Records in the UCS Database

For non-Open Media interactions, interaction records are normally created in the UCS Database without URS involvement.

For example, in the case of voice interactions:

- After an agent clicks Mark Done in Genesys Desktop, the UCS Database is updated. Users can also use the Genesys Desktop application to add and edit contact and contact history in the UCS Database as described in *Genesys Desktop Help*.

In the case of e-mail interactions, the UCS Database is updated as follows:

- When processing ordinary e-mails that arrive via the enterprise e-mail server or from a website via the Web API Server, E-mail Server Java stores the body of the interaction in the UCS Database (and sends the operational data to Interaction Server).

In the case of processing web media, such as chat:

- After a chat session ends, Chat Server writes the content of the chat session to the UCS Database.

Note: For more information on the UCS Database and the processing of e-mail and chat interactions, see the *eServices (Multimedia) 8.1 Deployment Guide*.

For Open Media interactions, no records are created in the UCS Database unless the Open Media UCS API or the Create Interaction object are used.

Use Case

One of the possible scenarios for using the Create Interaction object is as follows:

- The Media Server submits interactions to Interaction Server with the content of the interaction attached as User Data.
- A strategy executed by URS uses the Create Interaction object to create an interaction in the UCS Database and to store interaction-related data.
- When the interaction is routed to an agent, the Agent Application uses the UCS Database to obtain the content of the interaction in order to display it to the agent.

Recommendations

When interaction content is expected to be large, Genesys recommends:

1. Using the Create Interaction object as early as possible in a strategy.
2. Emptying the values of keys used for interaction content (if Create Interaction was processed successfully) after interaction data is stored in the UCS Database. This prevents large amounts of User Data from being passed between applications while large interactions are being processed. To empty keys, use the MultiAttach object to update attached data for these keys with empty values.

After all required (full) interaction data is stored in the UCS Database, emptying interaction data unimportant for routing in Interaction Server minimizes the amount of data transferred between Interaction Server, URS, Stat Server and certain Genesys Framework components. This can result in increased performance.

The Create Interaction object enables you to use the UCS Database for storing interaction content even if the Media Server is not integrated with UCS.

Returned Results for the Create Interaction Object

The results returned will be either an Event3rdServerResponse message or a fault message (Event3rdServerFault, see [page 151](#)). Any fault message returned uses the fault codes shown in [Table 37](#):

Table 37: Create Interaction Object Fault Codes

FaultCode value	FaultString value	Description	Action
201	Missing parameter name	One parameter is missing	Exit from strategy
202	Parameter ‘1’ and ‘2’ are not allowed	These two parameters cannot be present simultaneously	Exit from strategy

Table 37: Create Interaction Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
204	Incorrect value for parameter <parameter_name>, expected value 1 but was value 2	The specified parameter has an invalid value	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected message from the server's client	End of processing
512	Incorrect subtype for interaction <interaction_name>, expected type 1 but was type 2	Invalid interaction subtype	End of processing
701	Unexpected error exception message	An unexpected error happened	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry
732	Invalid Tenant <tenant_name>	The identifier for the tenant is invalid	Retry



Create Email Out

Use the Create Email Out object to create an outbound e-mail that can be sent to a customer as a reply to an inquiry or be used in an outbound e-mail campaign.

Create Email Out General Tab

Figure 118 shows an example completed properties dialog box for the Create Email Out object.

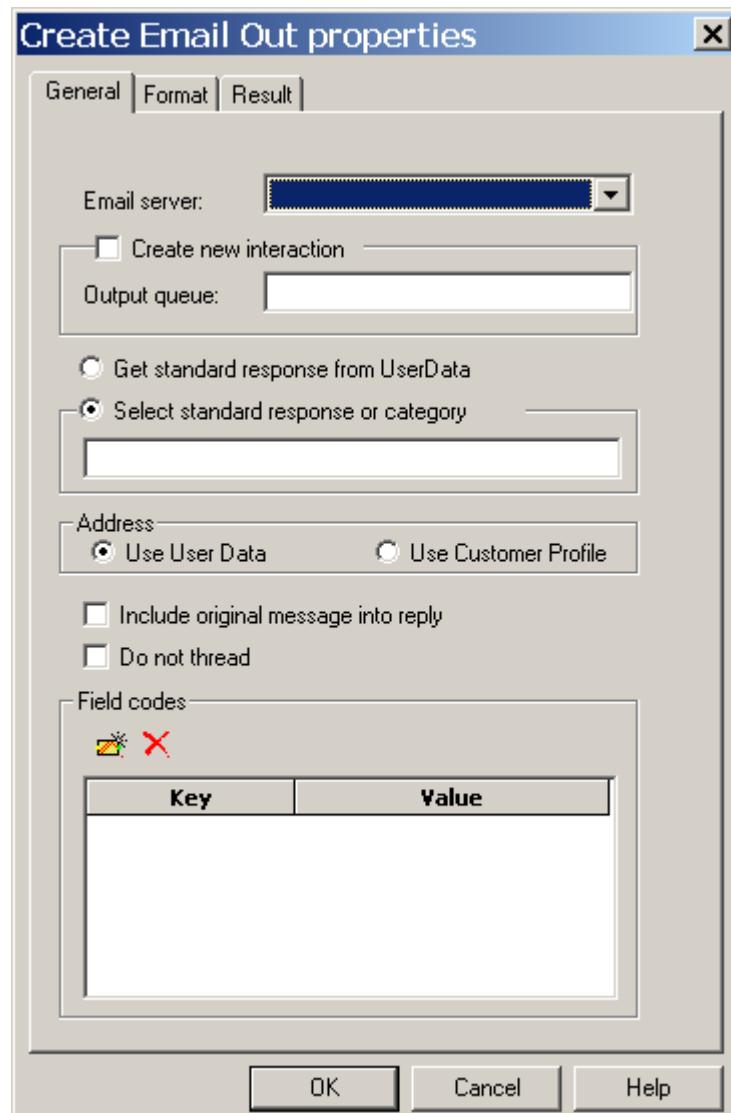


Figure 118: Create Email Out Properties Dialog Box

Use the information in [Table 36](#) to complete the General tab of the properties dialog box.

Table 38: Create Email Out, General Tab

Parameter	Description
Email Server	Click the down arrow and select an E-Mail Server Java Application from those configured in Configuration Manager.
Output queue	Required. Click in the Output queue text box to expand the business processes. Select the interaction queue where E-mail Server Java should place the outbound e-mail.
Create new interaction	Select this check box to have a new interaction record created in the UCS Database for this outbound e-mail.
Select source of standard response text	You can take the Standard Response text either from user data attached to the parent interaction or from any Standard Response in your Standard Response library. If you select the Select Standard Response or category radio button, click in the text box that appears to open a tree of available Standard Responses.
Address	Select the radio button that indicates from where (from user data attached to the parent interaction or the customer profile stored in the UCS Database) to take the address to which the e-mail is to be sent. If you choose to take the address from the database, you must place the Identify Contact object in the strategy before the Create Email Out object. For information on the Identify Contact object, see “Identify Contact” on page 237 .
Include original message into reply	Select this check box to have the text from the parent interaction copied into the outbound e-mail.
Do not thread	If selected, the outbound e-mail is not saved as a thread under an existing interaction (specified by under key ParentID in UserData) in the contact’s history in the UCS Database. By default this option is cleared.
Field codes	You can use this pane to assign values to any Field Code custom variables that may have been associated with the Standard Response in Knowledge Manager. See the <i>eServices (Multimedia) 8.1 User’s Guide</i> for more information on field codes. To assign field code values, click the new item button (). A new row appears under the Key and Value headings. Under Key, existing Field Code Custom Variables associated with the response appear for selection. Under Value, any existing strategy variables display for selection. Use the information in the “Field Codes and Custom Variables” on page 166 section below to edit the values. Note: You can also manually enter field codes.

Create Email Out Format Tab

The Create Email Out Format tab has only two fields:

- **From**—Select the appropriate address to appear in the From field of the outbound e-mail from the drop-down list of available e-mail addresses.
- **Subject**—Either enter a subject into the Subject text box or select the Use subject from SRL check box. If you select the check box, the e-mail subject is taken from the Standard Response you selected on the General tab.

Create Email Out Result Tab

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Use the information in Table 20 on [page 173](#) to complete the Result tab in the CreateEmailOut Properties dialog box.

Returned Results for the Create Email Out Object

The results returned will be either Event3rdServerResponse or Event3rdServerFault.

The content of the Event3rdServerResponse message is shown in [Table 39](#):

Table 39: Create Email Out Object Returned Results

Key	Description
NewInteractionId	The interaction ID of the new outbound e-mail.
SrlId	Global unique Standard Response identifier if it was found and used.

Any fault message returned uses the fault codes shown in [Table 40](#):

Table 40: Create Email Out Object Fault Codes

FaultCode value	FaultString value	Description	Action
105	No text found in Standard Response <SR_name>	The Standard Response exists but it contains no text	Exit from strategy
106	No ‘From/To’ address found	Either the From or the To address is missing	Exit from strategy
107	Too many ‘From/To’ addresses found	Either the From or the To field (or both) contain too many addresses	Exit from strategy
108	‘incorrect name’. Invalid ‘address’	The address or the domain is invalid	Exit from strategy
109	Standard response <SR_name> validity period ‘not yet started/expired’	The period during which the specified Standard Response is valid has not started or has ended	Exit from strategy
111	Standard response <SR_name> not found	This Standard Response does not exist in the database.	Exit from strategy
112	No Standard Response found in database	The database has no Standard Responses in it	Exit from strategy
201	Missing parameter name	One parameter is missing	Exit from strategy
202	Parameter ‘1’ and ‘2’ are not allowed	These two parameters cannot be present simultaneously	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
204	Incorrect value for parameter <parameter_name>, expected value 1 but was value 2	The specified parameter has an invalid value	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected message from the server’s client	End of processing

Table 40: Create Email Out Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
516	Interaction <interaction_name> already exists in database	This interaction already has been created in the database	End of processing
701	Unexpected error exception message	An unexpected error happened	Retry
710	Connection to database failed	Connection to database failed	Retry
711	Cannot get config of 'server'	E-Mail Server Java configuration error	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry
717	Config error: invalid option <option_name> in section <section_name>	Configuration error	Retry
718	Invalid custom field code key; <key_name> is a reserved key	Invalid custom field code key	Retry
732	Invalid Tenant <tenant_name>	The identifier for the tenant is invalid	Retry
733	No tenant declared in CME for application <app_name>	The application has no tenant specified	Retry
734	Application <app_name> has more than one associated tenant (in CME)	This application has more than one tenant specified	Retry
737	Request already in progress, request rejected	A second request was made for an interaction that already has a request in progress	Retry



Create Notification

Use the Create Notification object to create a notification e-mail that can be sent to a customer as a reply to an inquiry. This e-mail may itself contain the response to the inquiry or it may point the customer to the location of the information—for example, a page on the enterprise website.

Create Notification General Tab

Figure 119 shows an example completed properties dialog box for the Create Notification object.

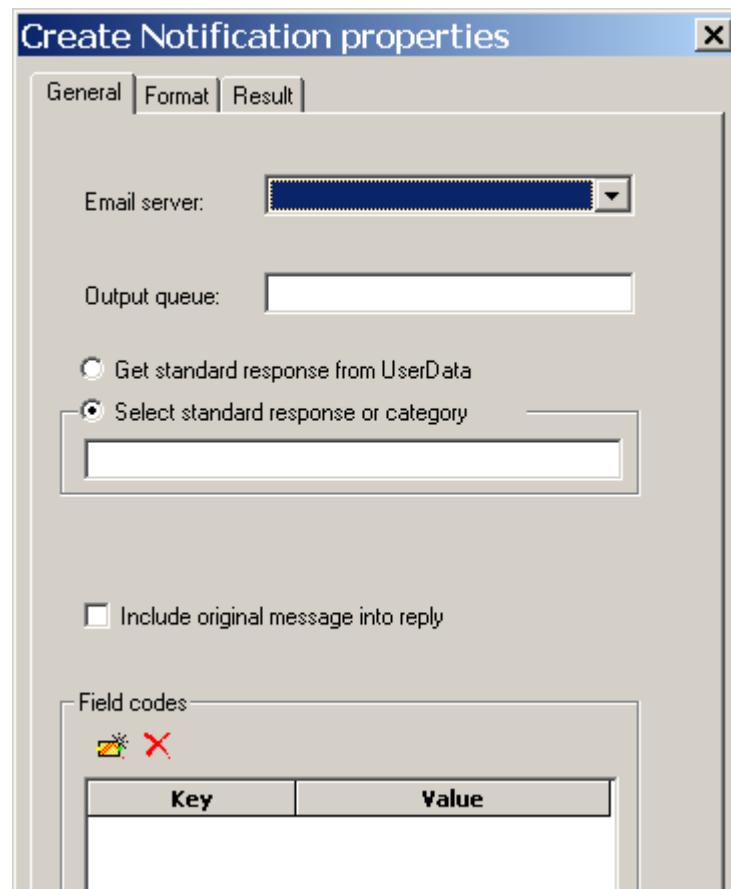


Figure 119: Create Notification Properties Dialog Box

Use the information in [Table 41](#) to complete the General tab of the properties dialog box.

Table 41: Create Notification, General Tab

Parameter	Description
Email Server	Click the down arrow and select an E-Mail Server Java Application from those configured in Configuration Manager.
Output queue	Required. Click in the Output queue text box to expand the business processes. Select the interaction queue where E-mail Server Java should place the notification e-mail.
Select source of standard response text	You can take the Standard Response text either from user data attached to the parent interaction or from any Standard Response in your Standard Response library. If you select the Select Standard Response or category radio button, click in the text box that appears to open a tree of available Standard Responses.
Address	Select the radio button that indicates from where (from user data attached to the parent interaction or the customer profile stored in the UCS Database) to take the address to which the e-mail is to be sent.
Include original message into reply	Select this check box to have the text from the parent interaction copied into the outbound e-mail.
Field codes	<p>You can use this pane to assign values to any Field Code custom variables that may have been associated with the Standard Response in Knowledge Manager. See the <i>eServices (Multimedia) 8.1 User's Guide</i> for more information on field codes.</p> <p>To assign field code values, click the new item button (). A new row appears under the Key and Value headings. Under Key, existing Field Code Custom Variables associated with the response appear for selection. Under Value, any existing strategy variables display for selection. Use the information in the “Field Codes and Custom Variables” on page 166 section below to edit the values.</p>

Create Notification Format Tab

The Create Notification Format tab has only two fields:

- **From**—Select the appropriate address to appear in the From field of the notification e-mail from the drop-down list of available e-mail addresses.
- **Subject**—Either enter a subject into the Subject text box or select the Use subject from SRL check box. If you select the check box, the e-mail subject is taken from the Standard Response you selected on the General tab.

Create Notification Result Tab

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Use the information in Table 20 on [page 173](#) to complete the Result tab in the Create Notification Properties dialog box.

Returned Results for the Create Notification Object

The results returned will be either Event3rdServerResponse or Event3rdServerFault.

The content of the Event3rdServerResponse message is shown in [Table 42](#):

Table 42: Create Notification Object Returned Results

Key	Description
NewInteractionId	The interaction ID of the new interaction.
SrlId	Global unique Standard Response identifier if it was found and used.

Any fault message returned uses the fault codes shown in [Table 43](#):

Table 43: Create Notification Object Fault Codes

FaultCode value	FaultString value	Description	Action
105	No text found in Standard Response <SR_name>	The Standard Response exists but it contains no text	Exit from strategy
106	No ‘From/To’ address found	Either the From or the To address is missing	Exit from strategy
107	Too many ‘From/To’ addresses found	Either the From or the To field (or both) contain too many addresses	Exit from strategy
108	‘incorrect name’. Invalid ‘address’	The address or the domain is invalid	Exit from strategy

Table 43: Create Notification Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
109	Standard response <SR_name> validity period 'not yet started/expired'	The period during which the specified Standard Response is valid has not started or has ended	Exit from strategy
111	Standard response <SR_name> not found	This Standard Response does not exist in the database.	Exit from strategy
112	No Standard Response found in database	The database has no Standard Responses in it	Exit from strategy
201	Missing parameter name	One parameter is missing	Exit from strategy
202	Parameter '1' and '2' are not allowed	These two parameters cannot be present simultaneously	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
204	Incorrect value for parameter <parameter_name>, expected value 1 but was value 2	The specified parameter has an invalid value	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected message from the server's client	End of processing
701	Unexpected error exception message	An unexpected error happened	Retry
710	Connection to database failed	Connection to database failed	Retry
711	Cannot get config of 'server'	E-Mail Server Java configuration error	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry

Table 43: Create Notification Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
717	Config error: invalid option <option_name> in section <section_name>	Configuration error	Retry
718	Invalid custom field code key; <key_name> is a reserved key	Invalid custom field code key	Retry
732	Invalid Tenant <tenant_name>	The identifier for the tenant is invalid	Retry
733	No tenant declared in CME for application <app_name>	The application has no tenant specified	Retry
734	Application <app_name> has more than one associated tenant (in CME)	This application has more than one tenant specified	Retry



Create SMS

Use the Create SMS object to create an outgoing message in a specially-designed format that can be sent via an E-Mail-to-SMS gateway. The gateway then transmits the message in Short Message Service (SMS) format, which is used to communicate with cell phones and personal digital assistants (PDAs).

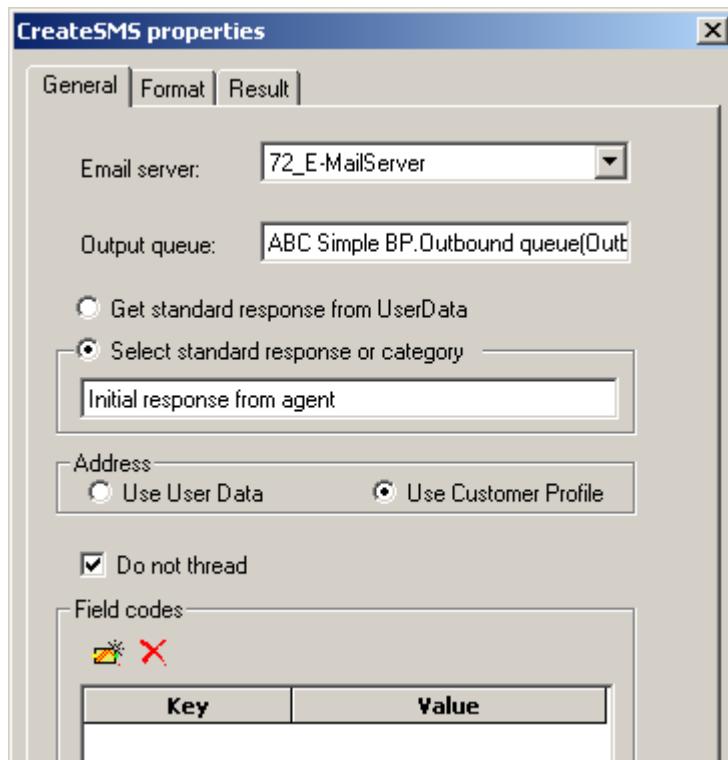
Genesys supports the Clickatell, sms2email.com, and SMS Gateway for Mdaemon gateways. Depending on the gateway you are using, the required message format differs. The messages are constructed in Knowledge Manager, as a form of Standard Response.

- For the settings required to create these messages, see the *Genesys eServices (Multimedia) 8.1 Knowledge Manager Help*.

Also review the documentation and the information on using your particular E-Mail-to-SMS gateway available on the gateway's website.

Create SMS General Tab

Figure 120 shows an example completed General tab in the properties dialog box for the Create SMS object.

**Figure 120: Create SMS Properties Dialog Box**

Use the information in [Table 44](#) to complete the General tab of the properties dialog box.

Table 44: Create SMS General Tab

Parameter	Description
Email Server	Click the down arrow and select an E-Mail Server Java Application from those configured in Configuration Manager.
Output queue	Required. Click in the Output queue text box to expand the business processes. Select the interaction queue where E-mail Server Java should place the notification e-mail.
Select source of standard response text	You can take the Standard Response text either from user data attached to the parent interaction or from any Standard Response in your Standard Response library. If you select the Select Standard Response or category radio button, click in the text box that appears to open a tree of available Standard Responses. The Standard Response you select must be constructed in a way that is compatible with the E-Mail-to-SMS gateway you are using. For an explanation of how to construct valid Standard Responses, see the Genesys eServices (Multimedia) 8.1 Knowledge Manager documentation.



Table 44: Create SMS General Tab (Continued)

Parameter	Description
Address	Select the radio button that indicates whether to take the address to which the e-mail is to be sent from user data attached to the parent interaction or from the customer profile stored in the UCS Database. <ul style="list-style-type: none"> • For Clickatell, select Use User Data. The user data should be configured with the key <code>_OutboundToAddress</code> and the value <code>sms@Messaging.clickatell.com</code> • For sms2email.com, select Use User Data. The user data should be configured with the key <code>_OutboundToAddress</code> and the value <code><any_mobile_number>@text.sms2email.com</code>. • For SMS Gateway for MDaemon, select Use User Data. The user data should be configured with the key <code>_OutboundToAddress</code>. The value can differ depending on the exact format you prefer. For detailed instructions, see the SMS Gateway for MDaemon website.
Do not thread	If you do not want the record of this SMS message to be stored as a thread under the parent interaction, select this check box. By default, this check box is cleared.
Field codes	You can use this pane to assign values to any Field Code custom variables that may have been associated with the Standard Response in Knowledge Manager. See the <i>eServices (Multimedia) 8.1 User's Guide</i> for more information on field codes. <p>To assign field code values, click the new item button (). A new row appears under the Key and Value headings. Under Key, existing Field Code Custom Variables associated with the response appear for selection. Under Value, any existing strategy variables display for selection. Use the information in the "Field Codes and Custom Variables" on page 166 section below to edit the values.</p> <p>Note: You can also manually enter field codes.</p>

Create SMS Format Tab

The Create SMS Format tab has only two fields:

- **From**—Select the appropriate address to appear in the `From` field of the notification e-mail from the drop-down list of available e-mail addresses. You can use any valid RFC address.
- **Subject**—Either enter a subject into the Subject text box or select the `Use subject from SRL` check box. If you select the check box, the e-mail subject is taken from the Standard Response you selected on the General tab.

Create SMS Result Tab

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Use the information in Table 20 on [page 173](#) to complete the Result tab in the Create SMS Properties dialog box.

Returned Results for the Create SMS Object

The results returned will be either Event3rdServerResponse or Event3rdServerFault.

The content of the Event3rdServerResponse message is shown in [Table 45](#):

Table 45: Create SMS Object Returned Results

Key	Description
NewInteractionId	The interaction ID of the new interaction.
SrlId	Global unique Standard Response identifier if it was found and used.

Any fault message returned uses the fault codes shown in [Table 46](#).

Table 46: Create SMS Object Fault Codes

FaultCode value	FaultString value	Description	Action
105	No text found in Standard Response <SR_name>	The Standard Response exists but it contains no text	Exit from strategy
106	No ‘From/To’ address found	Either the From or the To address is missing	Exit from strategy
107	Too many ‘From/To’ addresses found	Either the From or the To field (or both) contain too many addresses	Exit from strategy
108	‘incorrect name’. Invalid ‘address’	The address or the domain is invalid	Exit from strategy

Table 46: Create SMS Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
109	Standard response <SR_name> validity period 'not yet started/expired'	The period during which the specified Standard Response is valid has not started or has ended	Exit from strategy
111	Standard response <SR_name> not found	This Standard Response does not exist in the database.	Exit from strategy
112	No Standard Response found in database	The database has no Standard Responses in it	Exit from strategy
201	Missing parameter name	One parameter is missing	Exit from strategy
202	Parameter '1' and '2' are not allowed	These two parameters cannot be present simultaneously	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
204	Incorrect value for parameter <parameter_name>, expected value 1 but was value 2	The specified parameter has an invalid value	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected message from the server's client	End of processing
516	Interaction <interaction_name> already exists in database	This interaction has already been created in the database	End of processing
701	Unexpected error exception message	An unexpected error happened	Retry
710	Connection to database failed	Connection to database failed	Retry
711	Cannot get config of 'server'	E-Mail Server Java configuration error	Retry

Table 46: Create SMS Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
716	Server overloaded, request rejected	The server is overloaded	Retry
717	Config error: invalid option <option_name> in section <section_name>	Configuration error	Retry
718	Invalid custom field code key; <key name> is a reserved key	Invalid custom field code key	Retry
732	Invalid Tenant <tenant_name>	The identifier for the tenant is invalid	Retry
733	No tenant declared in CME for application <app_name>	The application has no tenant specified	Retry
734	Application <app_name> has more than one associated tenant (in CME)	This application has more than one tenant specified	Retry



Distribute Custom Event

Use the Distribute Custom Event object to define and distribute a custom reporting event to an Interaction Server. This object causes URS to generate a request to the Interaction Server for the method `DistributeCustomEvent`.

The Distribute Custom Event object enables you to generate your own custom reporting events that can be consumed by real-time or historical reporting to calculate or to generate more relevant and more accurate business reports than the reports based on standard reporting events.

The `DistributeCustomEvent` method does not use the current interaction user data. IRD does not send current user data to Interaction Server for this method.

Distribute Custom Event General Tab

Figure 121 shows the General tab in the properties dialog box for the Distribute Custom Event object.

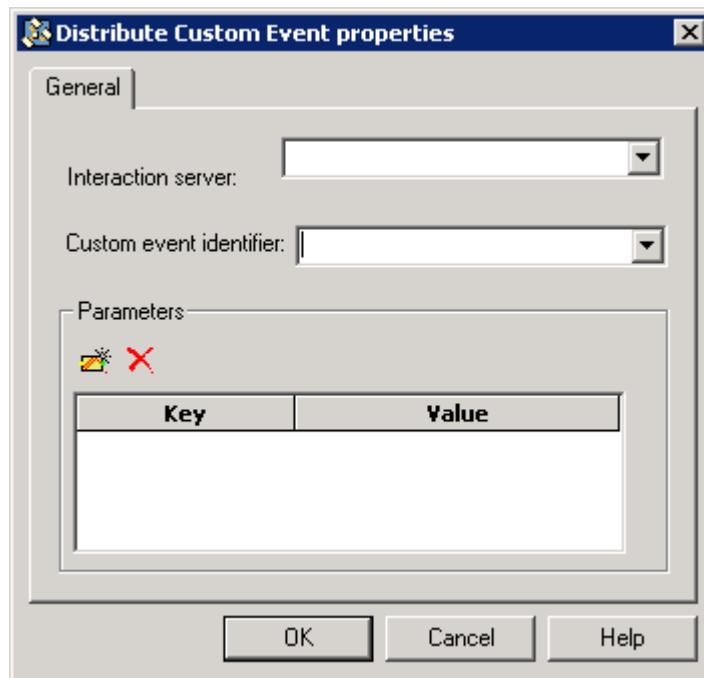


Figure 121: Distribute Custom Event Properties Dialog Box

Use the information in [Table 47](#) to complete the General tab of the properties dialog box.

Table 47: Distribute Custom Event, General Tab

Parameter	Description
Interaction Server	Select an interaction server from the Interaction server drop-down list, or type the name of the interaction server. Select/Type the server where you want this ESP request to be executed.
Custom Event Identifier	Select a custom event identifier from the Custom event identifier drop-down list, or type an <i>integer</i> to serve as the custom event identifier.
Parameters	You can use this pane to specify new custom event properties (which may or may not relate to the current interaction being processed by the strategy). To assign parameter values, click the new item button (). A new row appears under the Key and Value headings. Under Key, specify or select a reporting event property name. Under Value, specify or select a reporting event property value. This may be a constant, variable, or expression. The key-value list is inserted directly into the Parameters sublist of the ESP (similar to a generic ESP block).

Returned Results for the Distribute Custom Event Object

The results returned will be either `Event3rdServerResponse` (see [page 151](#)) or a fault message (`Event3rdServerFault`, see [page 151](#)).

Any fault message returned uses the fault codes shown in [Table 48](#):

Table 48: Distribute Custom Event New Interaction Object Fault Codes

FaultCode value	FaultString value	Description	Action
105	Invalid request data	The parameters list has not been attached. (This error might not be reproducible)	Exit from strategy
201	Missing parameter 'EventCustomId'	The only required parameter has not been specified	Exit from strategy
213	Invalid type of parameter <parameter_name>	The parameter is of an incorrect type	Exit from strategy
531	Invalid time format specified in one of the attributes	A string attribute can't be converted into a timestamp	End of processing
598	Custom event id parameter is out of range	Custom eventid is < 0, or > 9999.	End of processing



Find Interaction

Use the Find Interaction object to query Interaction Server for all or a number of interactions that correspond to specific conditions and listed in a specific order. This object causes URS to generate a request to Interaction Server for the method `FindInteractions`.

For implementation of complex scenarios where you need to make decisions regarding processing of one interaction based on data from another interaction, or if while processing one interaction you want to update another interaction, you need to be able to find the targeted interaction among the pool of all currently existing interactions in Interaction Server. The Find Interaction object accomplishes this. The interaction you are looking for may be a “parent interaction,” “interaction from the same customer,” and so on.



Find Interaction General Tab

[Figure 122](#) shows the dialog box that opens when you click the Find Interaction button, place the Find Interaction object in a strategy, and double-click to open its dialog box.

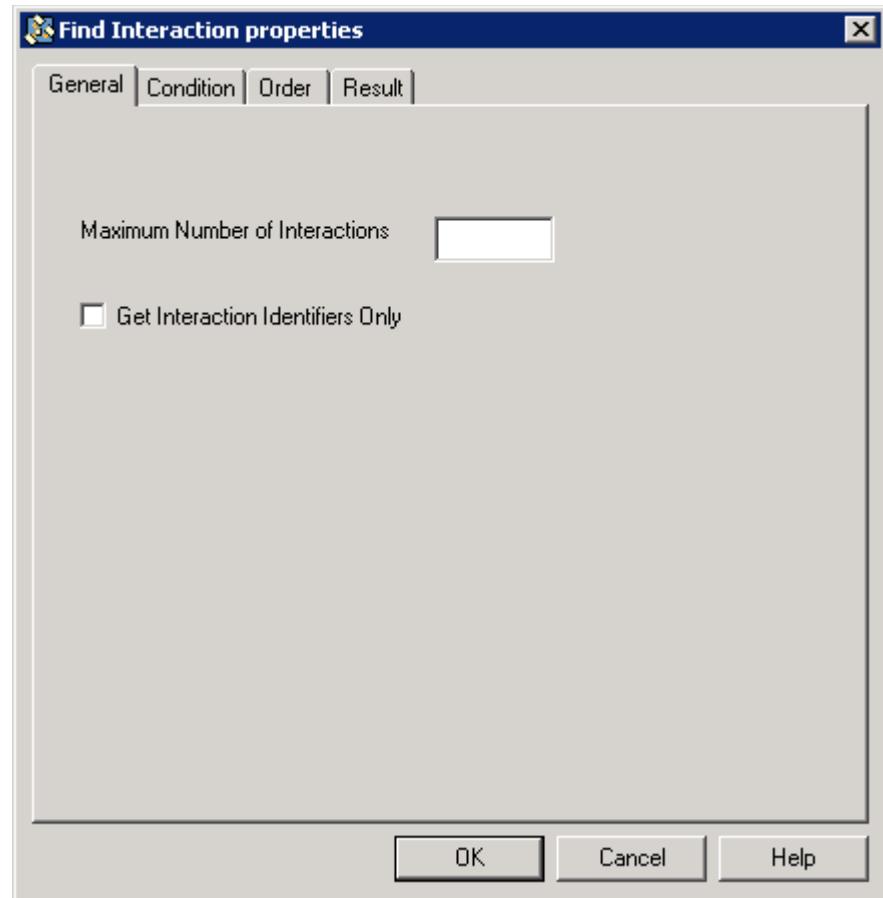


Figure 122: Find Interaction Dialog Box

Use the information in [Table 49](#) to complete the General tab in the Find Interaction Properties dialog box.

Table 49: Find Interaction Object, General Tab

Parameter	Description
Maximum Number of Interactions	Specify a positive integer value to limit the number of interactions returned in the result. If not specified, one interaction will be returned in the result.
Get Interaction Identifiers Only	Select this check box to receive only interaction ID in the results. If not selected, the results include an interactions list with full properties.

Find Interaction Condition Tab

Figure 123 shows the Condition tab of the Find Interaction object.

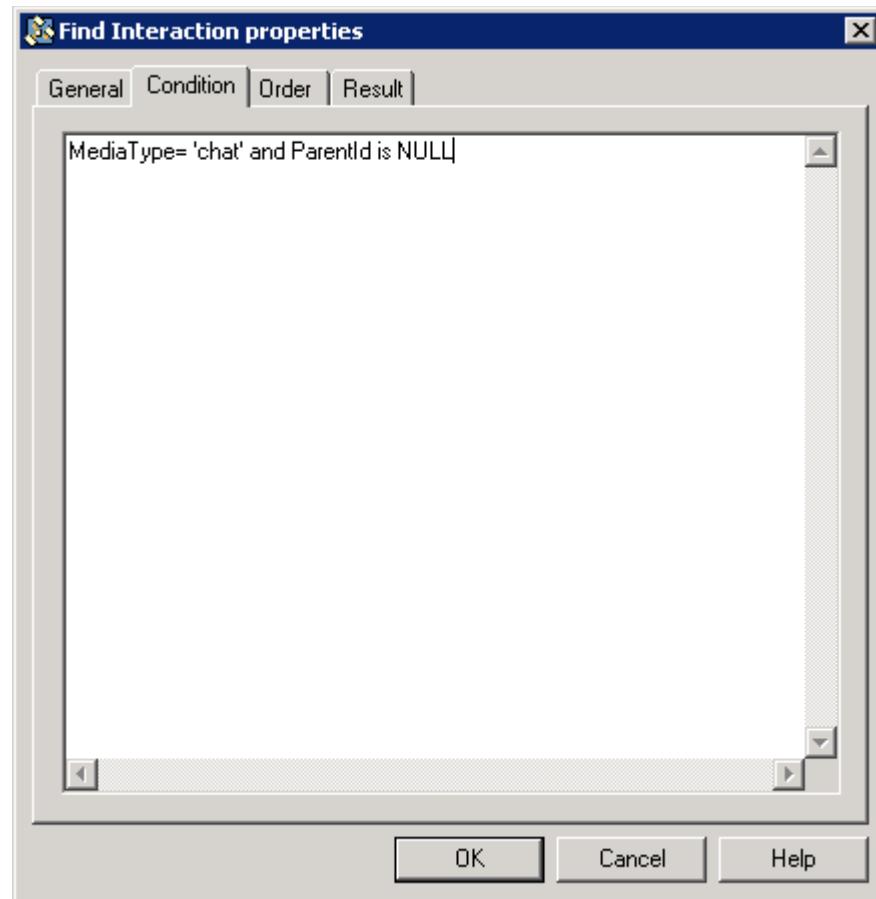


Figure 123: Find Interaction Dialog Box - Condition Tab

The Condition allows you to specify a WHERE clause to be used in a SQL statement to query interactions.

Note: If a Condition is not specified, the result list will have all interactions from this tenant.

Find Interaction Order Tab

Figure 124 shows the Order tab of the Find Interaction object.

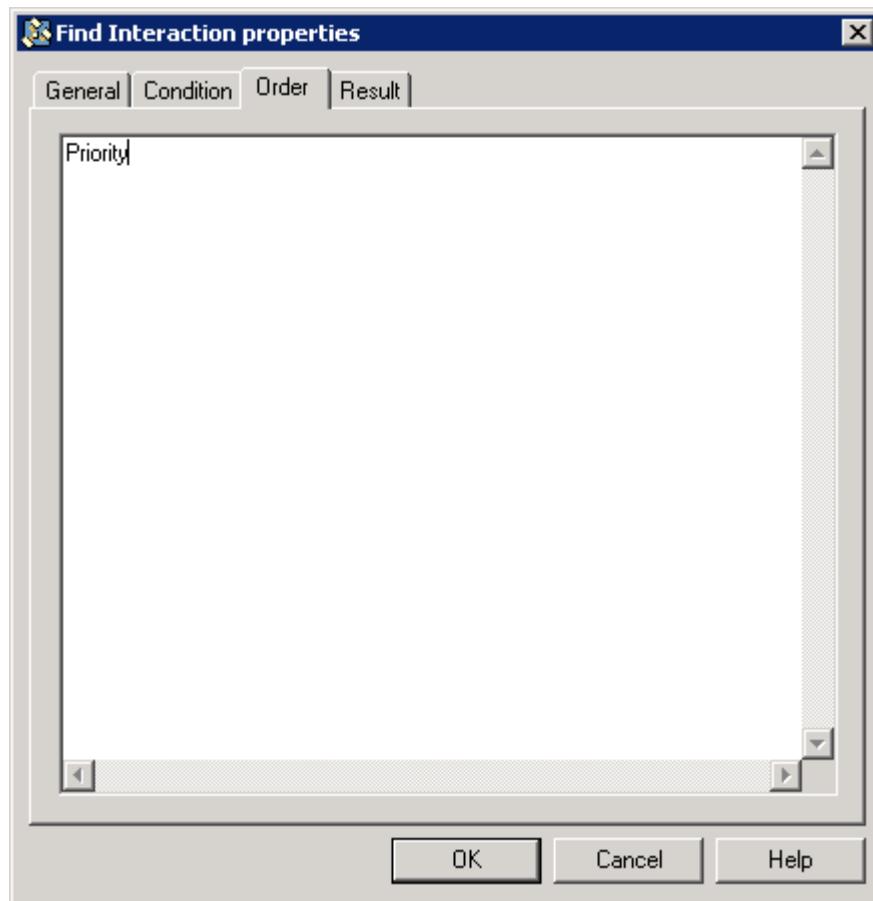


Figure 124: Find Interaction Dialog Box - Order Tab

The Order allows you to specify an ORDER clause to be used in a SQL statement to query interactions.

Note: If Order is not specified, the resulting interactions list will be sorted by received_at and then by id interaction properties. If a Condition was also not specified, the result list will have all interactions from this tenant.

Find Interaction Result Tab

[Figure 125](#) shows the Result tab of the Find Interaction object.

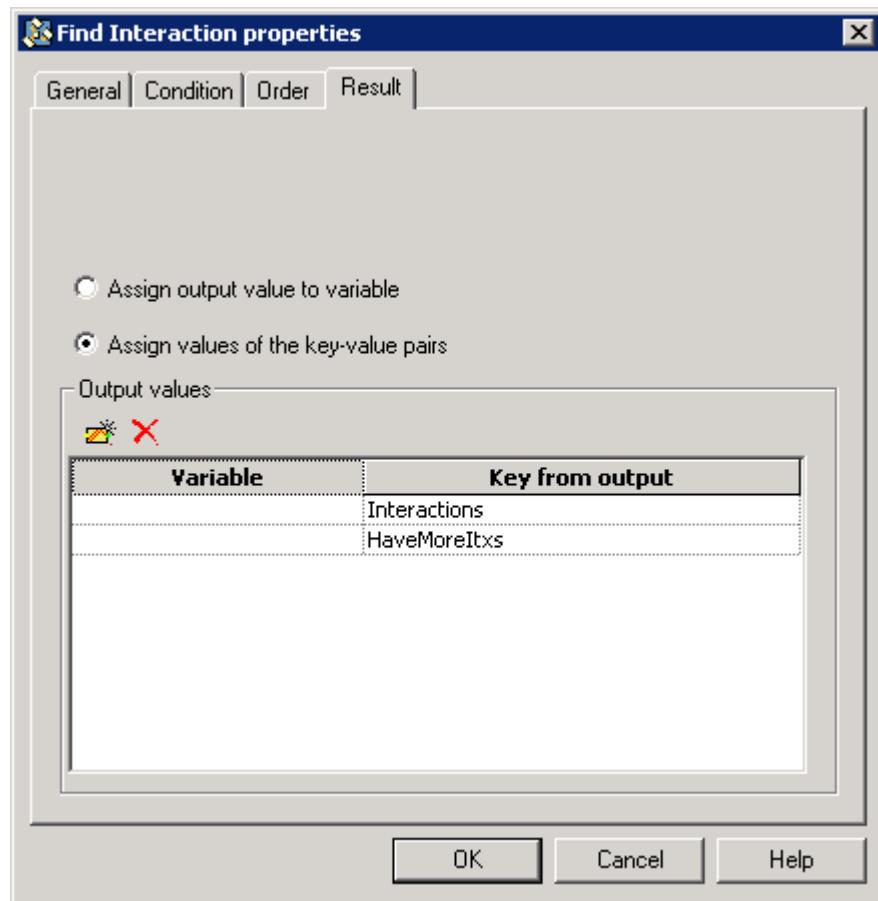


Figure 125: Find Interaction Dialog Box - Result Tab

Returned Results for the Find Interaction Object

The results returned will be an Event3rdServerResponse message. The content of the Event3rdServerResponse message is shown in [Table 50](#).

Table 50: Find Interaction Object - Content of Event3rdServerResponse

Key	Description
Interactions	List of interaction(s) (IDs only of full properties) found.
HaveMoreItxs	Indicates that there are more interactions that answer the requested condition but they were not returned in results.

Any fault message returned uses the fault codes shown in [Table 51](#).

Table 51: Find Interaction Object Fault Codes

FaultCode value	FaultString value	Description	Action
206	Unknown tenant	The tenant can not be found	
596	Failed to execute 'find' database request	An error occurred during the attempt to search the database	
597	Invalid condition or order in 'find' database request	An invalid condition or order SQL statement was specified that does not allow a successful search request to be completed	



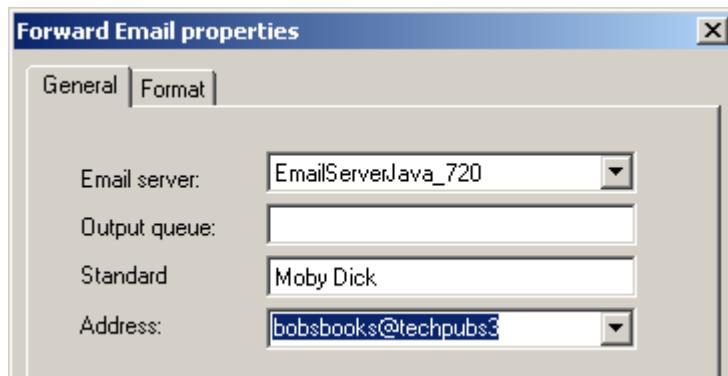
Forward E-Mail

Use the Forward E-Mail object to send a customer e-mail to an external address with the expectation of getting a response back, such as for agent collaboration. Use the Redirect E-Mail object (see [page 250](#)) if there you do not expect to receive a response. Once the external resource reply arrives as an inbound e-mail, use the Reply E-Mail From External Resource object (see [page 258](#)) to create the reply outbound e-mail.

The Forward E-Mail object is a request to E-mail Server Java for method `Forward` to create the e-mail to be forwarded. See *Universal Routing 7.6 Strategy Samples* for an example of how to use this object.

Forward E-Mail General Tab

[Figure 126](#) shows the dialog box that opens when you click the Forward E-Mail button, place the Forward E-Mail object in a strategy, and double-click to open its dialog box.

**Figure 126: Forward E-Mail Dialog Box**

The Address field enables you to select an E-mail Accounts Business Attribute that has been configured in Configuration Manager (see Figure 48 on [page 102](#)) to be used for routing.

Use the information in [Table 52](#) to complete the General tab in the Forward E-Mail Properties dialog box.

Table 52: Forward E-Mail Object, General Tab

Parameter	Description
E-mail Server	Click the E-mail Server down arrow and select the Application name in Configuration Manager for the E-mail Server Java that URS should notify (via Interaction Server). <ul style="list-style-type: none"> This field may be left empty. If empty, Interaction Server uses the first available E-mail Server Java named in its Connections list.
Output Queue	Required. Click in the Output queue text box to expand the business processes. Select the interaction queue where E-mail Server Java should place the e-mail to be forwarded. You must supply a value for this field; otherwise the strategy will not compile.
Standard	Use to manually select a Standard Response to be used for the forwarded e-mail so the recipient knows the reason for forwarding.
Address	Enter an e-mail address or select an E-mail Accounts Business Attribute from Configuration Manager (see Figure 48 on page 102). Each account has a real address in Annex\general\address. You must supply a value for this field.

Returned Results for the Forward E-Mail Object

The results returned will be either Event3rdServerResponse (see [page 151](#)) or a fault message (Event3rdServerFault, see [page 151](#)).

Any fault message returned uses the fault codes shown in [Table 53](#).

Table 53: Forward E-Mail Object Fault Codes

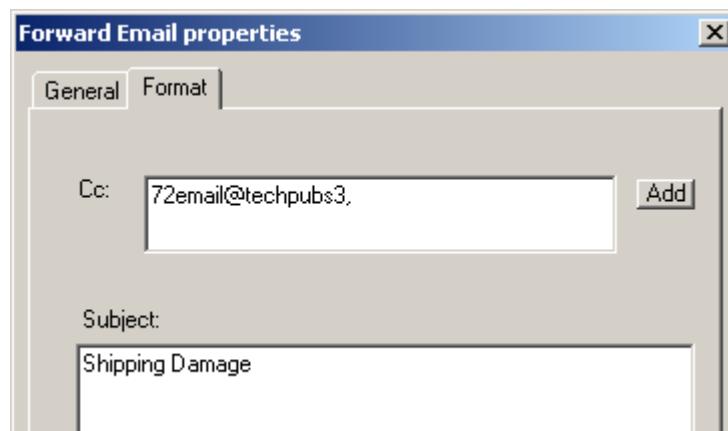
FaultCode value	FaultString value	Description	Action
105	No text found in Standard Response	This Standard Response is empty	Exit from strategy
106	No ‘From/To’ address found	Either the From or the To address is missing	Exit from strategy
107	Too many ‘From/To’ addresses found	Either the From or the To field (or both) contain too many addresses	Exit from strategy
108	‘incorrect name’. Invalid ‘address’	The address or the domain is invalid	Exit from strategy
109	Standard response <SR_name> validity period ‘not yet started/expired’	The period during which the specified Standard Response is valid has not started or has ended	Exit from strategy
201	Missing parameter name	One parameter is missing	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected message from the server’s client	End of processing
510	Object not found in database	Object is not found in database	End of processing
513	Interaction <interaction_name> failed. Manual recovery needed	Interaction is invalid	End of processing
701	Unexpected error exception message	An unexpected error happened	Retry
710	Connection to database failed	Connection to database failed	Retry

Table 53: Forward E-Mail Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
711	Cannot get config of 'server'	E-Mail Server Java configuration error	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry
717	Config error: invalid option <option_name> in section <section_name>	Configuration error	Retry
718	Invalid custom field code key; <key_name> is a reserved key	Invalid custom field code key	Retry
732	Invalid Tenant <tenant_name>	The identifier for the tenant is invalid	Retry
733	No tenant declared in CME for application <app_name>	The application has no tenant specified	Retry
734	Application <app_name> has more than one associated tenant (in CME)	This application has more than one tenant specified	Retry

Forward E-Mail Format Tab

You can customize the To, From, and Subject areas in the Standard Response e-mail. [Figure 127](#) shows the Format tab for the Forward E-Mail object.

**Figure 127: Forward E-Mail Object Format Tab**

Use the information in [Table 54](#) to complete the fields in the Format tab.

Table 54: Forward E-Mail Object Format Tab

Parameter	Description
Cc:	Click Add opposite Cc to select an address of anyone who should be copied when forwarding. The names listed are E-mail Accounts Business Attributes.
Subject:	If applicable, enter a new subject for the forwarded e-mail.



Identify Contact

Use the Identify Contact object to create a new contact, identify an existing one, and to update User Data associated with the interaction.

The Identify Contact object enables you to locate one or more contacts whose information is stored in the UCS Database including Last Called Agent information (see “Last Agent Routing” on [page 241](#) as well as the section on Media Contact Attributes for the Last Routed Agent feature in Chapter 4 of the *Genesys Desktop 8.1 Deployment Guide*). Contacts are located based on attributes attached to the interaction.

You can specify that you want UCS to return only a single matching contact or a list of contacts who match the specified search criteria. If you have UCS return only a single contact, all the contact’s attributes that are stored in the UCS Database are provided in the result.

If UCS cannot locate a matching contact, you can have it create a new contact that has the attributes attached to the interaction.

Identify Contact General Tab

[Figure 128](#) shows the General tab in the properties dialog box for the Identify Contact object.

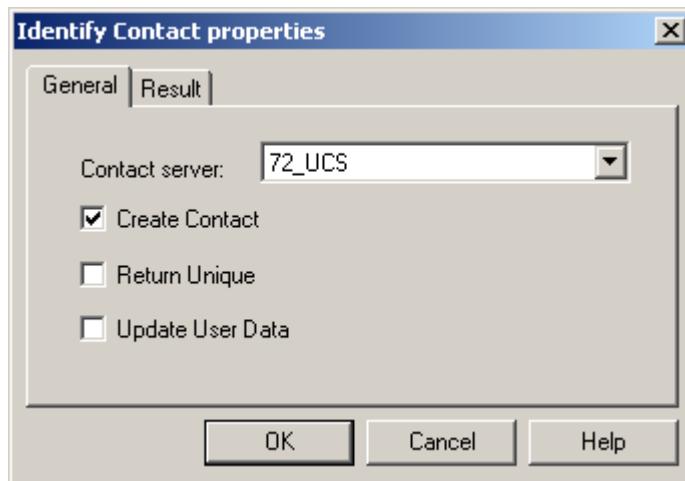


Figure 128: Identify Contact Properties Dialog Box

Use the information in [Table 55](#) to complete the General tab of the properties dialog box.

Table 55: Identify Contact General Tab

Parameter	Description
Contact Server	Click the down arrow and select a UCS Application from those configured in Configuration Manager.
Create Contact	Select this check box to have UCS create a new contact in the UCS Database if the search for an existing contact does not turn up a match.
Return Unique	Select this check box to have UCS return a single matching contact. If you leave this check box clear, UCS returns a list of matching contacts based on the set of contact attribute values you specify elsewhere in the strategy. If only a single contact is returned, UCS provides all attribute data available for that contact.
Update User Data	If the user data attached to the interaction differs from that stored in the UCS Database for the contact or contacts located with the Identify Contact object, UCS updates the database to include the values provided in the user data. See Table 57 for the user data that is returned if you select this option.

Identify Contact Result Tab

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Use the information in Table 20 on [page 173](#) to complete the Result tab in the Identify Contact Properties dialog box.

Returned Results for the Identify Contact Object

The results returned will be either Event3rdServerResponse (see [page 151](#)) or a fault message (Event3rdServerFault, see [page 151](#)).

The content of the Event3rdServerResponse message is shown in [Table 56](#):

Table 56: Identify Contact Object Returned Results

Key	Description
ContactCreated	If no matching contact exists in the UCS Database, Identify Contact can create a new contact. If one was created, the value returned is true. If not, the value is false.
NumberOfContacts Found	The number of contacts in the UCS Database that match the search criteria. If equal to 0, ContactIdList includes the newly created contact ID.
ContactIdList	A list of the contact of IDs of all contacts that match the search criteria you specified in the Identify Contact properties dialog box.

If you selected the Update User Data check box, the following user data is returned, as shown in [Table 57](#).

Table 57: Identify Contact Object Returned User Data

Key	Description
ContactId	Contact id, if a unique contact was found or created. Otherwise, this key has no value.
LastCalledAgent_EmployeeID	The employee ID of the agent who most recently handled an interaction with the contact.
LastCalledAgent_TimeStamp	The time of the most recent interaction with the contact.

Table 57: Identify Contact Object Returned User Data (Continued)

Key	Description
PreferredAgent_EmployeeID	The employee ID of the agent with which the contact would most like to interact.
EmailAddress	The contact's e-mail address.
PhoneNumber	The contact's phone number.
FirstName	The contact's first name.
LastName	The contact's last name.
<ContactAttributeName>	The name of a Contact's attribute as defined in the Configuration Layer ContactAttribute Enumerator (primary attribute only).

Any fault message returned uses the fault codes shown in [Table 58](#):

Table 58: Identify Contact Object Fault Codes

FaultCode value	FaultString value	Description	Action
201	Missing parameter name	One parameter is missing	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected message from the server's client	End of processing
510	Object not found in database	Object is not found in database	End of processing
701	Unexpected error exception message	An unexpected error happened	Retry
710	Connection to database failed	Connection to database failed	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry
730	No searchable attribute	No contact attribute for which to search	Retry

Last Agent Routing

You can use the Identify Contact object to return information about the last contacted agent separately for every media type.

As introduced in Universal Contact Server 7.0, storage of general Last Called Agent information (not media specific) is performed using the following Contact Attributes:

- `LastCalledAgent_EmployeeID`. This attribute stores the employee identifier (not media-specific) of the Last Called Agent. In the UCS API, it is defined as a STRING, not searchable, not sortable.
- `LastCalledAgent_TimeStamp`. This attribute stores the date/time (not media-specific) when the Last Called Agent was involved with this contact. In the UCS API, it is defined as a DATE, not searchable, not sortable. When used in an Identify Contact service called from URS, it is defined as a STRING (DATE type is not supported) following ISO-8601 format: `yyyy-MM-dd'T'HH:mm:ss'Z'`. For example, `2003-04-01T13:01:02Z` represents one minute and two seconds after 1:00 PM in the afternoon of 2003-04-01 (reference is GMT).
- `PreferredAgent_EmployeeID`. This attribute stores the “preferred agent”(not media-specific) employee identifier. In the UCS API, it is defined as a STRING, not searchable, not sortable.

To support Last Called Agent capability for each predefined `Media Type Business Attribute` in Configuration Manager, UCS automatically creates the following contact attributes:

- `LastCalledAgent_EmployeeID_<Media Type name>`. This attribute stores the employee identifier of the Last Called Agent for a specific media. In the UCS API, it is defined as a STRING, not searchable, not sortable.
- `LastCalledAgent_TimeStamp_<Media Type name>`. This attribute stores the date/time when the Last Called Agent was involved with this contact for a specific media. In the UCS API, it is defined as a DATE, not searchable, not sortable. When used in an Identify Contact service called from URS, it is defined as a STRING (as DATE type is not supported), following ISO-8601 format: `yyyy-MM-dd'T'HH:mm:ss'Z'`. For example, `2003-04-01T13:01:02Z` represents one minute and two seconds after 1:00 PM in the afternoon of 2003-04-01 (reference is GMT).

`PreferredAgent_EmployeeID_<Media Type name>`. This attribute is used to store the “preferred agent” employee identifier for a specific media. In UCS API, it is defined as a STRING, not searchable, not sortable.



MultiScreen

The MultiScreen object works similarly to the Screen object (see [page 263](#)). Both objects request Classification Server to filter interactions for specific words or word patterns based on a Screening Rule. The difference between

Screen and MultiScreen is that MultiScreen can use multiple Screening Rules, supplies additional return options, and does not require a conditional test in the strategy to determine whether a match occurred.

As described in “Category Structure” on [page 201](#), Screening Rules can be associated with classification categories. This gives Classification Server the possibility to return not only matched Screening Rules, but also any categories assigned to those Screening Rules. You can then use these categories to select a standard response or for segmenting interactions in the strategy.

MutiScreen General Tab

[Figure 129](#) shows the General tab of the MultiScreen Properties dialog box.

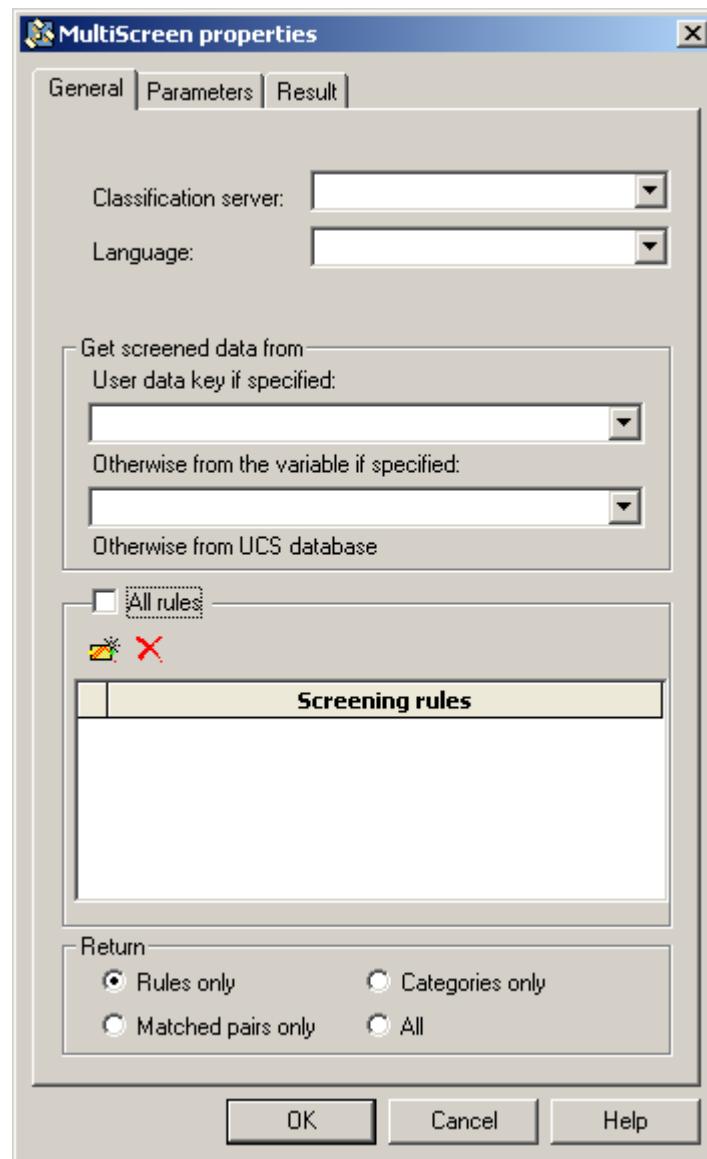


Figure 129: MultiScreen Properties Dialog Box, General Tab

Use the Screen Segmentation object ([page 384](#)) to process the results of the MultiScreen object.

Use the information in [Table 59](#) to complete the General tab.

Table 59: MultiScreen Object, General Tab

Parameter	Description
Classification	<p>Click the down arrow to select a Classification Server in the Configuration Server database.</p> <p>This field may be left empty. If empty, Interaction Server uses the first available Classification Server named in its Connections list.</p>
Language	<p>Click the down arrow to select the language of the incoming interaction. The selected language determines which Screening Rules are shown. You must select a language in order for the strategy to compile. Variables can be used for language, but the All Rules check box must be selected.</p>
Get screened data from	<p>You have three choices:</p> <ol style="list-style-type: none"> 1. User data key if specified. Use this option to have URS take the screened data from the user data attached to the interaction. 2. Otherwise from the variable if specified. Use this option to have URS use a variable for the screening rule name. You can enter the variable name or select it from the dropdown list. For a use case, see “Getting Screening Rules From Variables” on page 244. 3. Otherwise from the UCS Database. This is the default. Use this option to have URS take the screened data from the UCS Database, leave the text boxes empty.,
All Rules	<p>Select if you wish to apply all Screening Rules. If cleared, the down arrow for the drop-down list for selecting multiple Screening Rules appears as shown in Figure 129 on page 242.</p>
	<p>Click the down arrow to select a Screening Rule identifier (<code>credit card number</code> in Figure 129 on page 242). Only those Screening Rules with the option <code>Enabled=true</code> in the Configuration database are shown. If the Screening Rule identifier does not sufficiently describe the Screening Rules, you can:</p> <ul style="list-style-type: none"> • Look up the rule in Configuration Manager under Business Attributes > Screening Rules. • Look up the Screening Rule in Knowledge Manager, where it was originally defined. <p>You must select at least one Screening Rules in order for the strategy to compile.</p>
Return Rules only	<p>Select to return Screening Rule IDs (see Table 72 on page 267) when a match is found. See “Screening Rule Match Examples” on page 265.</p>

Table 59: MultiScreen Object, General Tab (Continued)

Parameter	Description
Matched pairs only	<p>Select to return matched pairs of Screening Rule IDs and specific strings of words in the e-mail that matched the Screening Rules.</p> <p>Classification Server can return specific strings that were matched during the screening process. For example a rule screening for credit card numbers could return the following key-value pair:</p> <p>"Key_credit_number" "1111-2222-3333-4444"</p>
Categories only	<p>Select to return the classification categories associated with the Screening Rules (see Figure 112 on page 199). This can be used later in the strategy to select a Standard Response. This is because a category can be associated with a Standard Response in Knowledge Manager, saved in the UCS Database, and the association then can be carried over to the Configuration Database, which makes it available to IRD strategies.</p>
All	<p>Select to return Screening Rule IDs, key-value pairs, and categories.</p>

Getting Screening Rules From Variables

You may wish to use this functionality with a strategy that uses multiple screen objects and variables containing Screening Rule names. These variables could be dynamically assigned in this strategy based on some sort of logic. For example, you could assign the value of the interaction's Customer Segment attribute to a variable. The UCS Database (Knowledge Manager) could be configured to have Screening Rules with names that the Customer Segment attribute could be equal to. So when this variable is assigned with a value of Gold, it could trigger the application of the Screening Rule with the same name in the strategy. The benefit of this functionality would be that the strategy containing these variables, once compiled, would dynamically know what Screening Rules to test against.

Note: When using variables for Screening Rules and Classification Server returns a “Rule didn’t match” error because the Screening Rule was not found, URS logic does not technically consider this to be an error. In this case, the interaction exits from the object’s green port.

How MultiScreen Processes Screening Requests

There are eight types of requests that the MultiScreen object can make to Classification Server. They are defined by the whether you select the **ALL rules** check box or leave it cleared and by which of the four radio buttons you select on the **Return** pane.

In order to return all possible categories and matched pairs for use on the agent desktop, the analysis doesn't stop at the first matched Screening Rules (although only the first one matched is returned in the `ScreenRuleName` and `Id` parameters of the Classification Server response). Analysis continues through either all rules in the system (if you select `All rules`) or the subset of rules specified in the `MultiScreen` properties dialog box (if `All rules` is cleared).

When the Order of Screening Rules Affects Results

If you select the `All rules` check box, the order in which the Screening Rules are placed can affect your results.

The lower the number, the higher priority the rule has (1 – higher, 100 – lower). Although the screening process analyzes all rules, only the first matched rule is returned in the `Id` and `ScreenRuleName` parameters of the response from Classification Server.

Therefore, rule 10 is analyzed before rule 20, and if rule 10 matches, rule 20 is not returned, even if it also matches.

When the Order of Screening Rules Does Not Affect Results

When the `All rules` check box is cleared, the order in which the Screening Rules are analyzed follows the order in which the rules are listed in the `MultiScreen` properties dialog box.

MultiScreen Parameters Tab

Figure 130 shows the Parameters tab of the MultiScreen Properties dialog box.

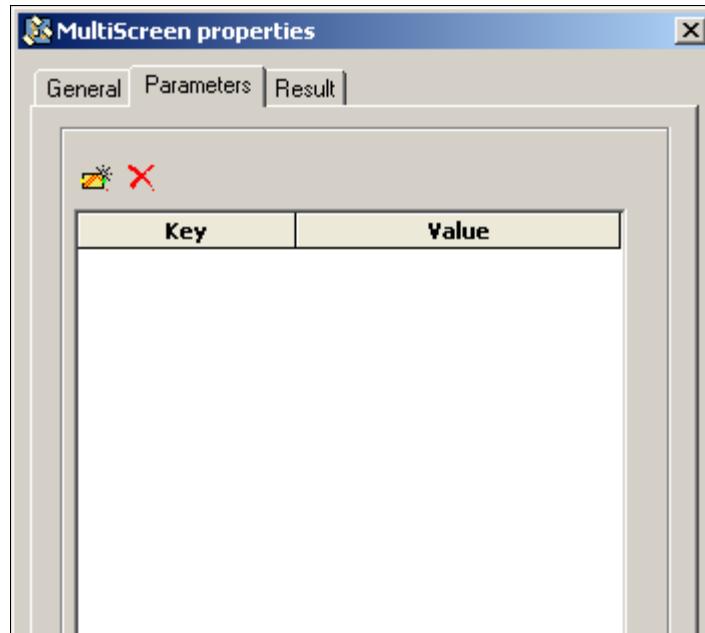


Figure 130: MultiScreen Properties Dialog Box, Parameters Tab

The Parameters tab allows you to provide any additional custom parameters with values that will be passed to Classification server “as is”.

MultiScreen Result Tab

Figure 131 shows the Result tab of the MultiScreen Properties dialog box.

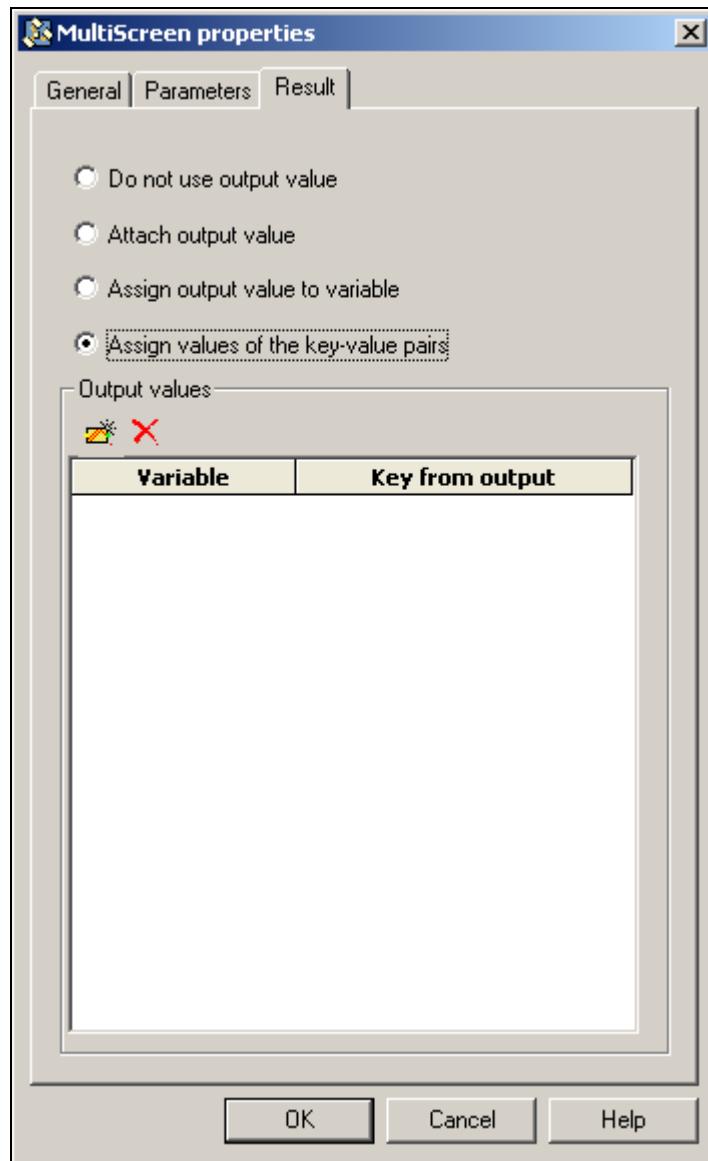
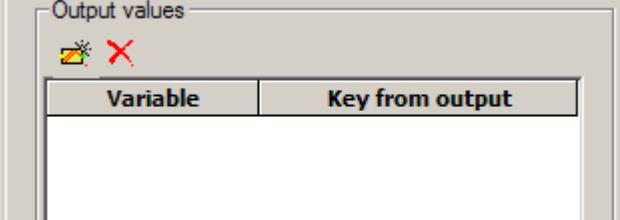


Figure 131: MultiScreen Properties Dialog Box, Result Tab

Use the information in the note on See “Screening Rule Match Examples” on [page 265](#) and the information in [Table 60](#) to complete the Result tab in the MultiScreen Properties dialog box.

Table 60: MultiScreen Object, Result Tab

Parameter	Description
Do not use output value	Select to have URS ignore the parameters data received from the server.
Attach output value	Select to attach returned values to the interaction User Data in the form of key-value pairs.
Assign output value to a variable	The typical way to process the result is to assign the results to a variable, which must already be defined (see “Variables in Objects and Expressions” on page 92).
Assign values of the key-value pair	Select to assign values in the returned array to of key-value pairs to a list of variables (see “Returned Results for the Screen Object” on page 268). If you select this option, the Result tab displays the Output Values pane. 
Output values	Click the button to add a new entry (). Under Variable (see Figure 139 on page 267), click in the field to display a down arrow. Click the down arrow and select the variable, which must already be defined (see “Variables in Objects and Expressions” on page 92). Under Key from output, use the key ScreenRuleMatch. See “Screening Rule Match Examples” on page 265 .

Note: *Universal Routing Strategy Samples* provides an example strategy that uses MultiScreen.

Returned Results for the MultiScreen Object

Note: Depending on the type of request, some parameters may not be present in the returned result. For example, if you selected Rules only in the General tab (see Figure 129 on [page 242](#)), then there will be no category list or matched strings list in the returned results. On the other hand, if you selected All, then Screening Rules, matched pairs, and categories will be present.

The screening response returned from Classification Server is in the form of an `Event3rdServerResponse` message (see [page 151](#)) or an `Event3rdServerFault` message (see [Table 62 on page 250](#)).

In the case of a match, the `Event3rdServerResponse` message (see [page 151](#)) will contain the information shown in [Table 61](#), with the value for `ScreenRuleMatch` being `true`:

Table 61: MultiScreen Object Event3rdServerResponse Message

Key	Description
ScreenRuleName	Key: ScreenRuleName Value: Name
Id	Key: Id Value: Actual identifier in Universal Contact Server Database
ScreenRuleMatch	Key: ScreenRuleMatch Value: true/false
CtgId	ID of the Category with the highest relevancy
CtgRelevancy	Relevancy of the Category presented in the CtgId
CtgName	Name of the Category with the highest relevancy.
Ctg_xxxx1	Category_id relevancy(75%)
Ctg_xxxN	Classification Server Results object
ScrKey_key1 Example: ScrKey_CreditCard	Key: "CreditCard" Value: "1111-2222-3333-4444" The keys are defined during Screening Rules creation in Knowledge Manager.
ScrKey_key2 Example: ScrKey_Technical_Support	Key: "Key_Technical_Support" Value: "Technical Support"

- If a match is not found, the `Event3rdServerResponse` message result includes value of `false` for the `ScreenRuleMatch` parameter. The interaction is then directed to the green port.
- If the message cannot be transmitted, and `Event3rdPartyFault` message returns one of the fault codes below.

Table 62: MultiScreen Fault Codes

FaultCode value	FaultString value	Description
501	Internal error in application. <additional information>	General internal error happened in Third-party application during request processing. For more details see application log.
502	Unexpected 3rdPty protocol message type=<value>	Protocol parser encountered unexpected protocol message from server's client.
504	Can't find interaction= <request message Id>	Server can't find specified interaction.
510	Can't find rule= <Rule Id>	Server can't find specified Screening Rule.
511	ucsapi exception: <description>, interaction= <request message Id>	Universal Contact Server API function exception.
512	Internal processing exception: <description>	Generic internal exception (out of memory, etc.).



Redirect E-Mail

Use this object to send a message to an external address without expecting any response or any further processing. The Redirect E-Mail object is a request to E-mail Server Java for method `Redirect` to create the redirected e-mail. Supports transfers to external resources, such as off-site experts or product specialists (knowledge workers). Use the Forward E-Mail object (see [page 250](#)) when you expect a response back.

Note: See *Universal Routing Strategy Samples* for an example strategy that uses the Redirect E-Mail object.

Redirect E-Mail General Tab

[Figure 132](#) shows the Redirect EMail properties dialog box.

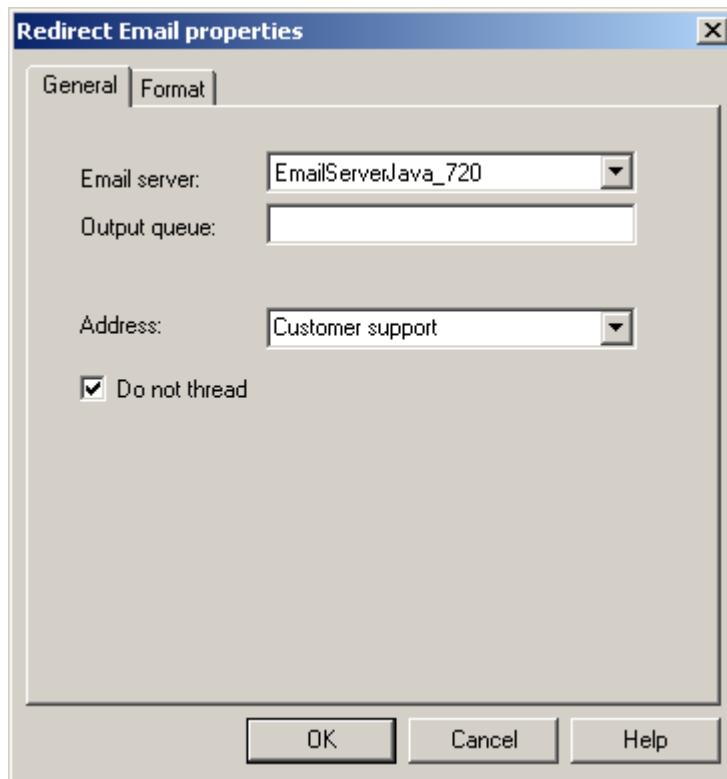


Figure 132: Redirect E-Mail Properties Dialog Box

Use the information in [Table 63](#) to complete the Redirect E-Mail properties dialog box.

Table 63: Redirect E-Mail Object, General Tab

Parameter	Description
E-mail server	Select the E-mail Server Java that will send the redirected e-mail. <ul style="list-style-type: none"> This field may be left empty. If empty, Interaction Server uses the first available E-mail Server Java named in its Connections list.
Output queue	Required. Click in the Output queue text box to expand the business processes. Select the interaction queue where E-mail Server Java should place the e-mail to be redirected. Queues configured in the Interaction Design window appear for selection. You must supply a value for this field; otherwise the strategy will not compile.
Address	Enter an e-mail address or select an E-mail Accounts Business Attribute from Configuration Manager (see Figure 48 on page 102) that can be used for routing. Each account has a real address in Annex\general\address. You must supply a value for this field.
Do not thread	Select this check box to prevent this interaction from being included in the interaction record for the redirected e-mail in the UCS Database.

Redirect E-Mail Format Tab

You can customize the To, From, and Subject areas in the Redirect E-Mail object. Figure 133 shows the Format tab for the Redirect E-Mail object.

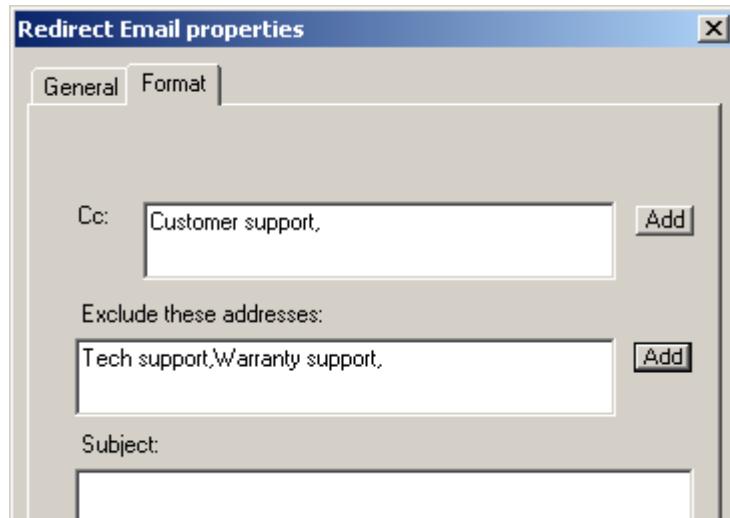


Figure 133: Redirect E-Mail Object Format Tab

Use the information in Table 64 to complete the fields in the Format tab.

Table 64: Redirect E-Mail Object Format Tab

Parameter	Description
Cc:	Click Add opposite Cc to select one or more addresses to which this redirected e-mail should be copied. All names listed are E-mail Accounts Business Attributes.
Exclude these addresses:	The Exclude these addresses pane lists addresses found in the original e-mail. Click Add to select addresses to exclude from the redirected e-mail. Click Variable if the addresses are contained in a variable. Can contain a comma-separated list of domains, aliases, variables, and e-mail addresses.
Subject:	If applicable, add a new Subject for the redirected e-mail.

Returned Results for the Redirect E-Mail Object

The results returned will be either Event3rdServerResponse (see [page 151](#)) or a fault message (Event3rdServerFault, see [page 151](#)).

Any fault message returned uses the fault codes shown in [Table 65](#):

Table 65: Redirect E-Mail Object Fault Codes

FaultCode value	FaultString value	Description	Action
106	No ‘From/To’ address found	Either the From or the To address is missing	Exit from strategy
107	Too many ‘From/To’ addresses found	Either the From or the To field (or both) contain too many addresses	Exit from strategy
108	‘incorrect name’. Invalid ‘address’	The address or the domain is invalid	Exit from strategy
201	Missing parameter name	One parameter is missing	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected message from the server’s client	End of processing
510	Object not found in database	Object is not found in database	End of processing
513	Interaction <interaction_name> failed. Manual recovery needed	Interaction is invalid	End of processing
701	Unexpected error exception message	An unexpected error happened	Retry
710	Connection to database failed	Connection to database failed	Retry
711	Cannot get config of ‘server’	E-Mail Server Java configuration error	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry
717	Config error: invalid option <option_name> in section <section_name>	Configuration error	Retry

Table 65: Redirect E-Mail Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
718	Invalid custom field code key; <key_name> is a reserved key	Invalid custom field code key	Retry
732	Invalid Tenant <tenant_name>	The identifier for the tenant is invalid	Retry
733	No tenant declared in CME for application <app_name>	The application has no tenant specified	Retry
734	Application <app_name> has more than one associated tenant (in CME)	This application has more than one tenant specified	Retry



Render Message Content

Use the Render Message Content object to create message content using text from either a message box of the strategy object, strategy variable, User Data or Standard Response. This object causes URS to generate a request to Universal Contact Server for the method `RenderMessageContent`.

In some situations (for example, outbound SMS or chat prompt), you may need to be able to create the message text inside an IRD strategy based on interaction context data, a standard response, or other data. Render Message Content provides this capability. The message may include customer name, estimated wait time, or other data as a parameter. The Render Message Content object will take the parameterized text (or standard response) and its parameter values and render the final text that will be sent to the customer.

Render Message Content General Tab

[Figure 134](#) shows the dialog box that opens when you click the Render Message Content button, place the Render Message Content object in a strategy, and double-click to open its dialog box.

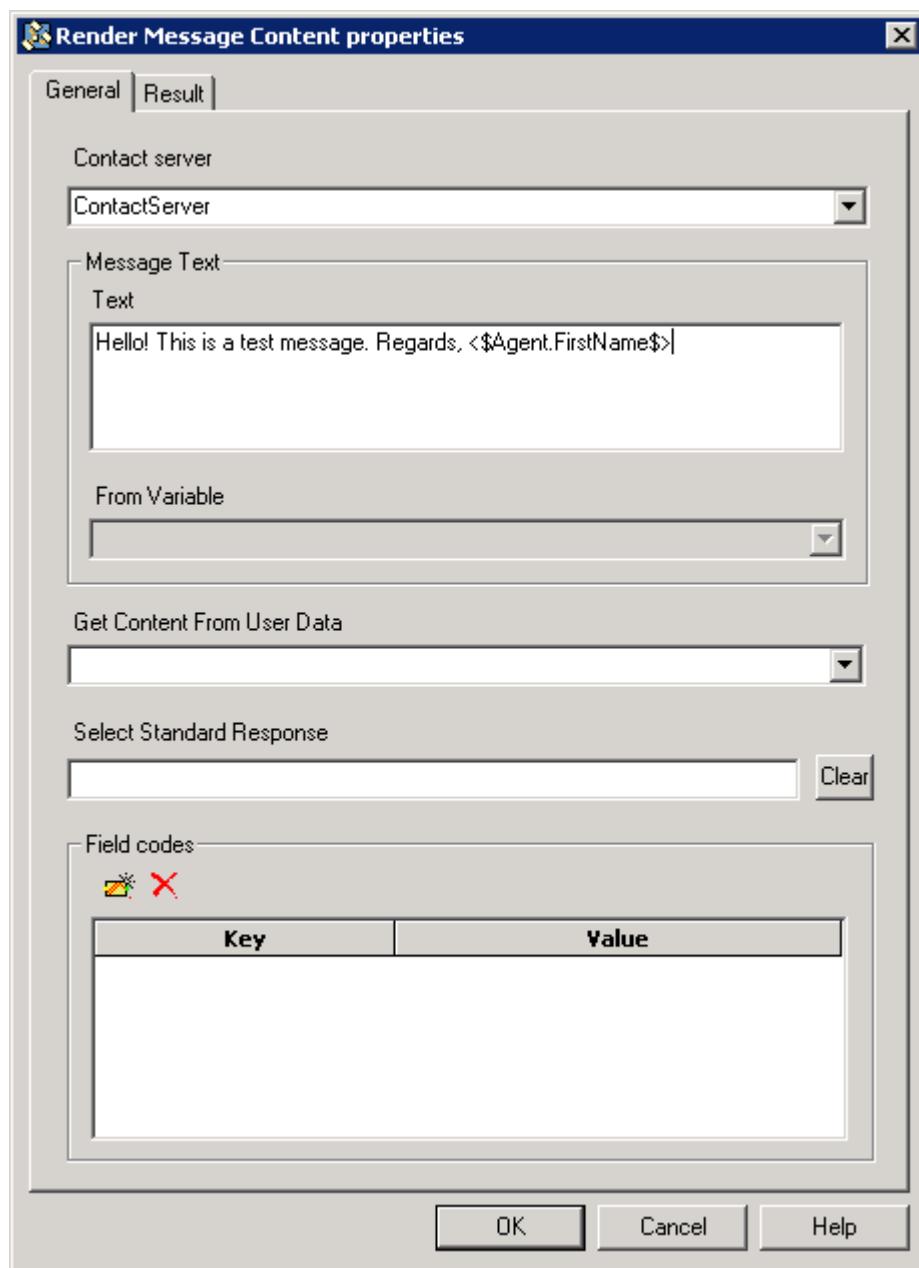


Figure 134: Render Message Content Dialog Box

Use the information in [Table 66](#) to complete the General tab in the Render Message Content Properties dialog box.

Table 66: Render Message Content, General Tab

Parameter	Description
Contact Server	This is a drop-down list with available UCS servers' names. Select a server where you want this ESP request to be executed.
Message Text	Enter text that needs to be rendered in this text box. The text should contain field codes (otherwise the ESP request might not make sense).
From Variable	This is a drop-down list with available variables. If text in the <code>Message Text</code> box above is not specified, then you must choose a variable containing the text that needs to be rendered here. A variable needs to be defined in the strategy and contain some text.
Get Content From User Data	If neither the <code>Message Text</code> box nor the <code>Get Content from Variable</code> box has specified content, you must use either this editable drop-down list with available variables that could contain a User Data key, or manually enter the key. This key in User Data contains text that needs to be rendered.
Select Standard Response	If neither of the above-mentioned controls is used, then you must use this drop-down list. The drop-down control will show available category trees – only approved, not expired, standard responses will be shown. Choose a standard response to render.
Field Codes	This is the same control that the Autoresponse object has, and it is used for the same purpose – to specify the custom values for custom field codes that may be used in the text source.

Render Message Content Result Tab

Configuring the Result Tab

[Figure 135](#) shows the Result tab of the Render Message Content object.

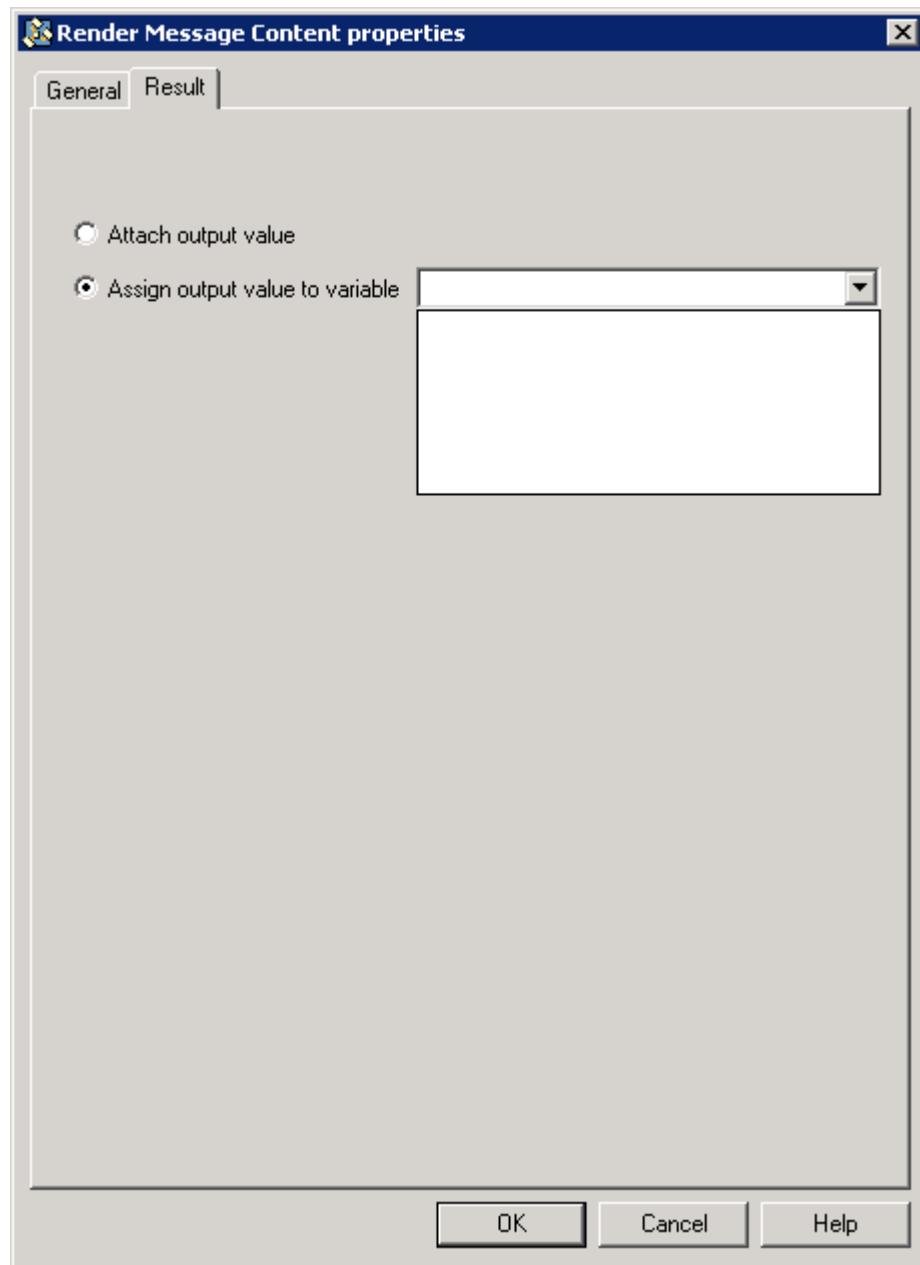


Figure 135: Render Message Content Dialog Box - Result Tab

- Select `Attach Output Value` if you want to attach returned values to the interaction. This is the default.
Select `Assign Output Value to Variable` if you want to assign any returned output values to a variable. (The variable must already have been defined.) When you select this option, a drop-down menu appears so you can select the variable.

Returned Results for the Render Message Content Object

The results returned will be an Event3rdServerResponse message. The content of the Event3rdServerResponse message is shown in [Table 67](#).

Table 67: Render Message Content Object - Content of Event3rdServerResponse

Key	Description
Text	Rendered plain text
StructuredText	Rendered rich formatted text
Subject	Subject of the Standard Response

Any fault message returned uses the fault codes shown in [Table 68](#).

Table 68: Render Message Content Object Fault Codes

FaultCode value	FaultString value	Description	Action
105	No text found in Standard Response <SR_name>	The Standard Response exists but it contains no text	Define Standard Response
502	"{0} not found in database"	Fieldcode or Text entity was not found in Database.	Define Fieldcode
701	Unexpected error exception message	An unexpected error happened	Retry
718	Invalid custom field code key; <key name> is a reserved key	Invalid custom field code key	Retry



Reply E-mail From External Resource

Note: *External resource* is a name for any object outside the contact center. It maybe an external agent or another contact center. Configure external e-mail addresses as E-Mail Accounts Business Attributes in Configuration Manager.

This object works with the Forward E-Mail object (see [page 233](#)). It takes the resulting External Resource Reply inbound e-mail as input, extracts the external resource reply text from it, creates a customer reply outbound e-mail, and submits the e-mail to Interaction Server using the specified interaction queue. For an example strategy, see *Universal Routing 7.6 Strategy Samples*.

Background

An inbound e-mail can be forwarded to an external resource (any e-mail address) using either the Forward E-Mail object in a routing strategy as described on [page 233](#)) or via the Genesys Agent Desktop (using the Forward to External Resource feature). The external resource then replies to this e-mail (using regular e-mail software, such as MS Outlook).

At that point, the following occurs:

- The reply e-mail comes back into Genesys eServices to E-mail Server
- E-mail Server Java identifies the e-mail as an External Resource Reply, and then submits it to Interaction Server.
- Interaction Server submits the e-mail to URS according to URS's own policies and queues.

A routing strategy containing the Reply E-Mail from External Resource object is then used. It extracts the agent reply from the External Resource Reply e-mail, and creates an e-mail for the customer (the External Resource Reply message is formatted a special way to allow this process to occur).

The final outbound e-mail customer reply can be either sent directly to the customer via the Send E-Mail object (see [page 269](#)) in the routing strategy, go to an agent for QA Review, or go as a draft to the original agent who forwarded the message.

Reply E-Mail from External Resource General Tab

[Figure 136](#) shows the properties dialog box associated with the Reply E-Mail From External Resource object.



Figure 136: Reply E-Mail from External Resource Dialog Box

Use the information in [Table 69](#) to complete the General tab in the Reply E-Mail From External Resource dialog box.

Table 69: Reply E-Mail From External Resource, General Tab

Parameter	Description
E-mail Server	Select the E-mail Server Java to handle the External Resource Reply inbound e-mail containing the draft reply text from the collaborating agent or expert.(see page 258).
Output queue	Required. Click in the Output queue text box to expand the business processes. Select the interaction queue where E-mail Server Java should place the outbound e-mail that contains the reply to be sent to the customer. You must supply a value for this field; otherwise the strategy will not compile.
Include original message into reply	Select this check box if the reply e-mail should attach the original e-mail.

Format Tab You can customize the To, From, and Subject areas in the Reply E-Mail. The Reply E-Mail From External Resource object uses the same Format tab as the Acknowledgement and Autoresponse objects (see Figure 94 on [page 169](#)). For information on completing the fields, see Table 18 on [page 170](#).

Returned Results

The results returned will include the agent's reply or be a fault message (Event3rdServerFault, [page 151](#)) using the fault codes listed in [Table 70](#):

Table 70: Reply E-Mail from External Resource Object Fault Codes

FaultCode value	FaultString value	Description	Action
106	No 'From/To' address found	Either the From or the To address is missing	Exit from strategy
107	Too many 'From/To' addresses found	Either the From or the To field (or both) contain too many addresses	Exit from strategy
108	'incorrect name'. Invalid 'address'	The address or the domain is invalid	Exit from strategy
201	Missing parameter name	One parameter is missing	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
204	Incorrect value for parameter <parameter_name>, expected value 1 but was value 2	The specified parameter has an invalid value	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected message from the server's client	End of processing
510	Object not found in database	Object is not found in database	End of processing
512	Incorrect subtype for interaction <interaction_name>, expected type 1 but was type 2	Invalid interaction subtype	End of processing
513	Interaction <interaction_name> failed. Manual recovery needed	Interaction is invalid	End of processing

Table 70: Reply E-Mail from External Resource Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
701	Unexpected error exception message	An unexpected error happened	Retry
710	Connection to database failed	Connection to database failed	Retry
711	Cannot get config of 'server'	E-Mail Server Java configuration error	Retry
714	Can't stop processing of interaction <interaction_name>	Cannot stop this interaction	Retry
715	Connection to <server_name> closed	A connection to the specified server has closed	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry
717	Config error: invalid option <option_name> in section <section_name>	Configuration error	Retry
718	Invalid custom field code key; <key_name> is a reserved key	Invalid custom field code key	Retry
732	Invalid Tenant <tenant_name>	The identifier for the tenant is invalid	Retry
733	No tenant declared in CME for application <app_name>	The application has no tenant specified	Retry
734	Application <app_name> has more than one associated tenant (in CME)	This application has more than one tenant specified	Retry
913	Supplied MimeMessage object does not contain an embedded Id	The message does not contain an embedded ID	Retry
914	No parent found for interaction <interaction_name>	Cannot find the parent interaction for this child interaction	Retry

Table 70: Reply E-Mail from External Resource Object Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
915	No reply text found in interaction <interaction_name>	This reply email contains no text	Retry
916	Mismatch between Id <ID_1> and interaction parent Id <ID_2>	The embedded ID and the parent ID do not match	Retry
917	No well constructed message <ID>	The specified message is not well constructed	Retry
918	Can't open the MimeMessage contained in the reply <ID>	Cannot open the MimeMessage in the specified reply	Retry



Screen

Use the Screen object to request Classification Server to filter (screen) interactions for specific words or patterns of words based on the Screening Rules sent by URS (see [Figure 137](#)).

Edit Screening Rule

Name:	Wrong transaction amount		
<input checked="" type="checkbox"/> Enabled	Order: 10		
Use these addresses			
customer_support tech_support warranty_support	Add		
<input type="checkbox"/> Exact address match			
<input checked="" type="radio"/> AND <input type="radio"/> OR			
Use pattern			
Choose and add regular expressions and operators between them:			
Find("")	Add	&&	Add
Find("transaction",true) && Find("amount",true) && (Find("wrong",true) Find("different",true) Find("doesn't match",true)			
Search for pattern in message's		<input checked="" type="checkbox"/> Subject	<input checked="" type="checkbox"/> Body
<input type="checkbox"/> Merge sources checked above		<input checked="" type="checkbox"/> Header	
Categories			
Add	Root category	Category	Relevancy
	Financial service	Errors in transactions	75
Test against whole database			
Test messages			
Add new	Subject		
Delete			
OK		Cancel	

Figure 137: Example Knowledge Manager Screening Rule

Notes: To obtain all available screening functionality, Genesys strongly recommends that you use the MultiScreen object (see [page 241](#)) instead of the Screen object. The Screen object should be used only for backward compatibility; for example, for 7.0 backward compatibility.

When you first start creating routing strategies, you may wish to use the Screen object instead of the Classify object if your site's classification categories are not yet defined or if your site has not purchased the Content Analysis option required by the Classify object.

Screening Rule Match Examples

The Screening Rule identifier is returned in the parameters if the interaction was matched with the rule. If parameter with the key `Id` is returned equal to `""` from Classification Server, this indicates that the message did not match the rule.

The typical way to process the result is to assign the screening result to a variable and then manipulate the interaction further based on the results assigned to the variable. For example:

- You write the results to a variable and the variable is equal to:
`ScreenRuleName: ""|Id: ""|ScreenRuleMatch: false|return:ok`
In this case, the message did not match the rule.
- You write the results to a variable and the variable is equal to:
`00001a05F5U900QW: TestDSL|ScreenRuleName: TestDSL|Id: 00001a05F5U900QW|ScreenRuleMatch: true|return:ok`
In this case, the message matched the rule.

Another typical way to use the data would be to attach all the parameters to the interaction's User Data. Then, further in the strategy, this User Data can be analyzed to make routing decisions.

Screen General Tab

Before completing the `Screen` dialog box, become familiar with your site's Screening Rules and corresponding Screening Rule identifiers. These are initially defined in Knowledge Manager and then carry over (refreshed every 10 seconds) to Configuration Manager, where they appear under `Business Attributes > Screening Rules` (see [Figure 48 on page 102](#)). Once the carryover occurs, IRD's `Screen` object dialog box displays the Screening Rules.

[Figure 138](#) shows the `General` tab in the `Screen` properties dialog box.

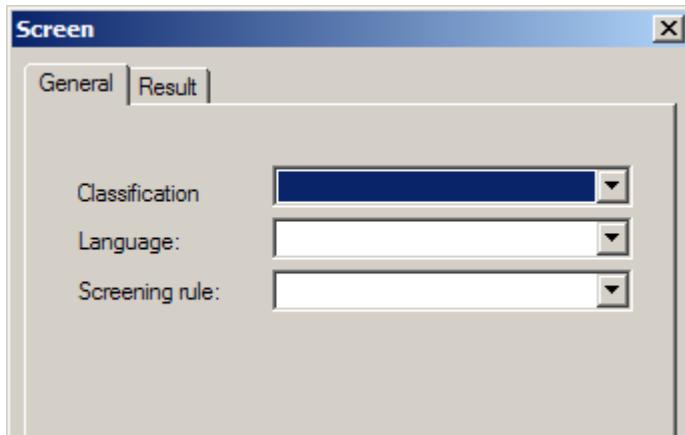


Figure 138: Screen Object Dialog Box

Use the information in [Table 71](#) to complete the General tab.

Table 71: Screen Object, General Tab

Parameter	Description
Classification	Click the down arrow to select a Classification Server in the Configuration Server database. <ul style="list-style-type: none">• This field may be left empty. If empty, Interaction Server uses the first available Classification Server named in its Connections list.
Language	Click the down arrow to select the language of the incoming interaction. The selected language determines the Screening Rules shown. You must select a language in order for the strategy to compile.
Screening rule	Click the down arrow to select a Screening Rule identifier (Name in Figure 137 on page 264). Only those Screening Rules with the option <code>Enabled=true</code> in the Configuration database are shown. If the Screening Rule identifier does not sufficiently describe the Screening Rules, you can: <ul style="list-style-type: none">• Look up the rule in Knowledge Manager (Edit Screening Rule dialog box). If you want to change the value of this attribute, you must do it in Knowledge Manager, not Configuration Manager.• If additional detail is needed, look up the Screening Rule in Knowledge Manager, where it was originally defined. You must select a Screening Rule in order for the strategy to compile.

Screen Result Tab

[Figure 139](#) shows the Result tab in the Screen properties dialog with an example entry.

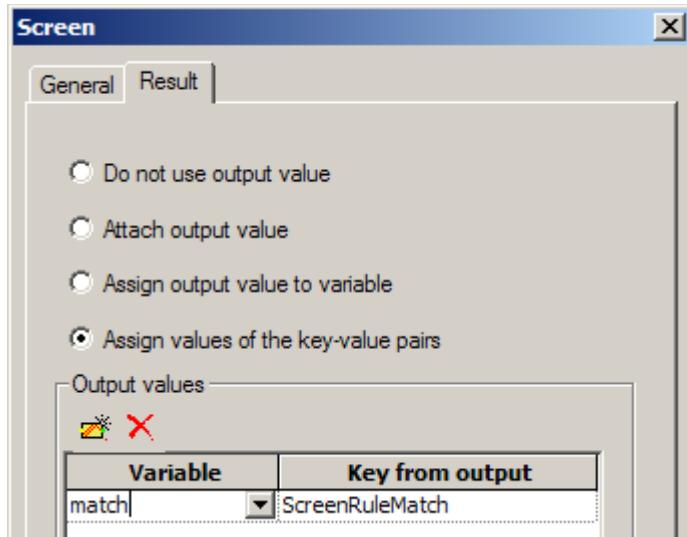


Figure 139: Screen Object Dialog Box, Result Tab

Screening Response

The screening response returned from Classification Server is in the form of an `Event3rdServerResponse` message (see [page 151](#)).

- In the case of a match, the `Event3rdServerResponse` message (see [page 151](#)) will contain the information shown in [Table 72](#), with the value for `ScreenRuleMatch` being `true`:

Table 72: Screen Object Event3rdServerResponse Message

Key	Description
Actual UCS DB ID	Key: Actual identifier in Universal Contact Server database. Value: Name
ScreenRuleName	Key: ScreenRuleName. Value: Name
Id	Key: Id. Value: Actual identifier in Universal Contact Server database
ScreenRuleMatch	Key: ScreenRuleMatch. Value: true/false

- If a match is not found, the `Event3rdServerResponse` message result includes value of `false` for the `ScreenRuleMatch` parameter. The interaction is then directed to the green port.

Use the information in the note on “Screening Rule Match Examples” on [page 265](#) and the information in [Table 73](#) to complete the Result tab in the Screen properties dialog box.

Table 73: Screen Object, Result Tab

Parameter	Description
Do not use output value	Select this radio button to have URS ignore the parameters data received from the server.
Attach output value	Select to attach returned values to the interaction User Data in the form of key-value pairs.
Assign output value to a variable	The typical way to process the result is to assign the results to a variable (see “Screening Rule Match Examples” on page 265). Select this radio button to assign any returned output values to an already-configured variable which must already be defined (see “Variables in Objects and Expressions” on page 92).
Assign values of the key-value pair	Click to assign values in the returned array of classification pairs to a list of variables (see “Returned Results for the Screen Object” on page 268). If you select this option, the Result tab displays the Output Values pane shown in Figure 139 on page 267 .
Output values	Click the button to add a new entry (). Under Variable (see Figure 139 on page 267), click in the field to display a down arrow. Click the down arrow and select the variable, which must already be defined (see “Variables in Objects and Expressions” on page 92). Under Key from output, use the key ScreenRuleMatch. See “Screening Rule Match Examples” on page 265 .

Returned Results for the Screen Object

The results returned will be either an Event3rdServerResponse message (see Table 72 on [page 267](#)) or a fault message (Event3rdServerFault) using the fault codes listed in [Table 74](#):

Table 74: Screen Object Fault Codes

FaultCode value	FaultString value	Description
501	Internal error in application with additional information	General internal error happened in Third-party application during request processing. For more details see application log.
502	Unexpected third party protocol message type with value	Protocol parser encountered unexpected protocol message from server's client.
504	Can't find interaction with request message ID	Server cannot find specified interaction.
510	Can't find rule with rule ID	Server cannot find specified Screening Rule.
511	Universal Contact Server API exception with description and message ID	UCS API function exception.

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.



Send E-Mail

Use this object to send an e-mail waiting in a queue that was previously created with the Acknowledgement, Autoresponse, Chat Transcript, Forward E-Mail, Redirect E-Mail, Reply E-Mail from External Resource, or Create Email Out object.

Send E-Mail General Tab

[Figure 140](#) shows the Send E-Mail properties dialog box.

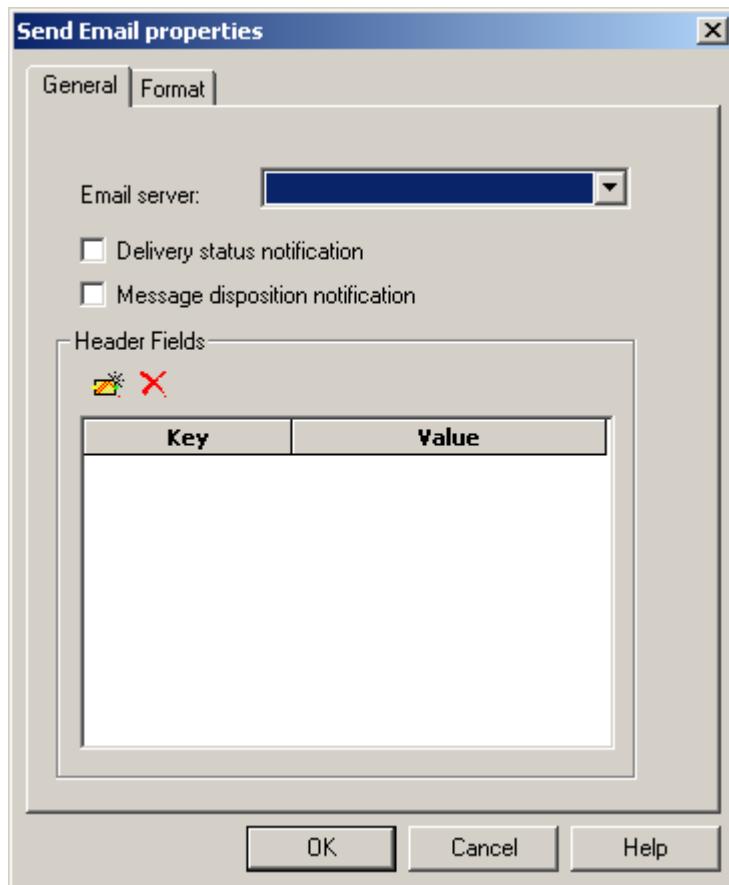


Figure 140: Send E-Mail Properties Dialog Box

Use the information in [Table 75](#) to complete the General tab of the Send E-Mail properties dialog box.

Table 75: Send E-Mail Properties Dialog Box, General Tab

Parameter	Description
E-mail server	<p>Select the E-mail Server Java that will send e-mail from the dropdown list.</p> <ul style="list-style-type: none"> This field may be left empty. If empty, Interaction Server uses the first available E-mail Server Java named in its Connections list.
Delivery status notification	<p>Check this box for the e-mail to include a request for a return message indicating whether and how the original e-mail was delivered.</p> <ul style="list-style-type: none"> If one of the SMTP servers involved in the transport of the original e-mail fails to deliver it, the return message comes into the system with subtype InboundNDR. It contains no additional information. If the original e-mail is successfully delivered, the return message comes into the system with subtype InboundReport. It uses attached data to indicate delivery statuses such as delayed, delivered, relayed, and so on. For details see the “E-mail Server Java: Advanced Topics” section of the “Ongoing Administration and Other Topics” chapter of the <i>eServices (Multimedia) 8.1 User's Guide</i>.
Message disposition notification	<p>Check this box for the e-mail to include a request for a return message indicating what happened to the original e-mail after it was delivered; for example, whether it was displayed, printed, deleted without displaying, and so on. The return message comes into the system with subtype InboundDisposition, and it provides the delivery status in attached data. For details see the “E-mail Server Java: Advanced Topics” section of the “Ongoing Administration and Other Topics” chapter of the <i>eServices (Multimedia) 8.1 User's Guide</i>.</p>
Header Fields	<p>Click the add button () to add a new header field.</p> <p>Under Key (see Figure 140 on page 270), enter the name of the new header field.</p> <p>Under Value, select a variable that contains the value or enter the value. Examples:</p> <p>X-modified: Thu, 29 Jun 2006 11:22:54 +0200 X-spam: spam_filter_name</p> <p>Warning: E-mail Server Java returns an error if:</p> <ul style="list-style-type: none"> One of the following (reserved by E-mail Server Java) are present: AddedHeaderList (from GUI parameters) and _AddedHeaderList (from UserData). One of the following (reserved by E-mail Server Java) is used for the header name: Return-Path, Received, Message-Id, Resent-Date, Date, Resent-From, From, Reply-To, Sender, To, Cc, Bcc, Subject, In-Reply-To, Resent-Message-Id, Errors-To, MIME-Version, Content-Type, Content-Transfer-Encoding, Content-MD5, Content-Length, or Status.

Attaching Header Fields to User Data

To attach new header fields to the interaction User Data, add the prefix `Header_` to the keys that are being attached to the User Data. If you proceed in this fashion, all clients that access User Data know that these particular keys come from the header. For more information, see the chapter on Interaction Properties, Other Properties section, in the *eServices (Multimedia) 8.1 User's Guide*.

Returned Results for the Send E-Mail Object

The results returned will be either a response message (`Event3rdServerResponse`, see [page 151](#)) or a fault message (`Event3rdServerFault`, see [page 151](#)). Any fault message returned uses the fault codes in [Table 76](#).

Table 76: Send E-Mail Object General Tab Fault Codes

FaultCode value	FaultString value	Description	Action
201	Missing parameter name	One parameter is missing	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
510	Object not found in database	Object is not found in database	End of processing
701	Unexpected error exception message	An unexpected error happened	Retry
710	Connection to database failed	Connection to database failed	Retry
711	Cannot get config of 'server'	E-Mail Server Java configuration error	Retry
712	Error during SMTP authentication	SMTP authentication error has occurred	Retry
713	Unknown host <host_name>	The mail server host is unknown	Retry
910	No valid sender found in <interaction_ID>	No valid From address in the e-mail	Route to agent or supervisor

Table 76: Send E-Mail Object General Tab Fault Codes (Continued)

FaultCode value	FaultString value	Description	Action
911	No valid recipients found in <interaction_ID>	No valid To address in the e-mail	Route to agent or supervisor
912	Bad address(es): <address(es)>	At least one address in the e-mail is invalid	Route to agent or supervisor

Send E-Mail Format Tab

You can customize the `To`, `From`, and `Subject` areas in the response e-mail to the customer. [Figure 141](#) shows the Format tab for the Send E-Mail object.

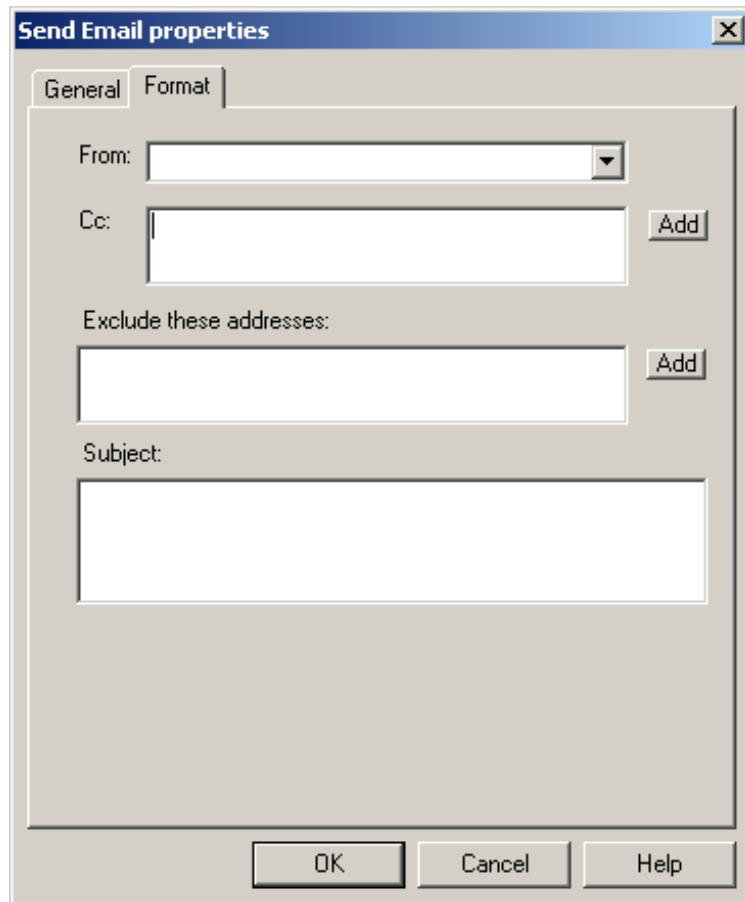


Figure 141: Send E-Mail Object, Format Tab

Use the information in [Table 77](#) to complete the Format tab.

Table 77: Send E-Mail Object Format Tab

Parameter	Description
From	If applicable, click the From: down arrow and select a From address for the e-mail. Addresses in the drop-down list were previously entered as E-mail Accounts under Business Attributes in Configuration Manager (see Figure 48 on page 102). Example: address3@domain.com. Can contain an e-mail address, variable or alias.
Cc	If applicable, click Add opposite Cc and select an address to copy into the e-mail. Addresses in the drop-down list were previously entered as E-mail Accounts under Business Attributes in Configuration Manager (see Figure 48 on page 102). Example: address2@domain.com;a3@domain.com. Can contain a list of e-mail addresses separated by a semi-colon as shown above. Can also contain variables or aliases.
Exclude these addresses	The Exclude these addresses text box lists addresses found in the original e-mail. Click Add and select addresses from the list in the Select Addresses dialog box to exclude them from the e-mail to be sent. Click Variable in the Select Addresses dialog box if the addresses are contained in a variable. Can contain a comma-separated list of domains, aliases, variables, and e-mail addresses.
Subject	If applicable, enter a new subject for the e-mail.



Stop Interaction

Sends a `RequestRouteCall(RouteTypeStop)` message to Interaction Server.

If you select the **Notify** check box, it also notifies UCS or a third party server (through Interaction Server) that interaction processing has stopped for a particular strategy within a business process.

If you are notifying a third-party server, you can specify the service and the method name, if necessary. If you are notifying UCS, use the default service (`Ixn`) and method name (`StopProcessing`).

When used in a strategy, generates a Stop node in a business process.

When to Use Stop Interaction

When a strategy is part of a business process (see [page 149](#)), always finish the strategy (and each strategy branch) with one of these routing objects: Stop Interaction, Queue Interaction (see [page 320](#)), Route Interaction (see [page 322](#)), or Workbin (see [page 348](#)).

Stop Interaction General Tab

Figure 142 shows the dialog box that opens when you click the Stop Interaction button and place the object in your strategy.

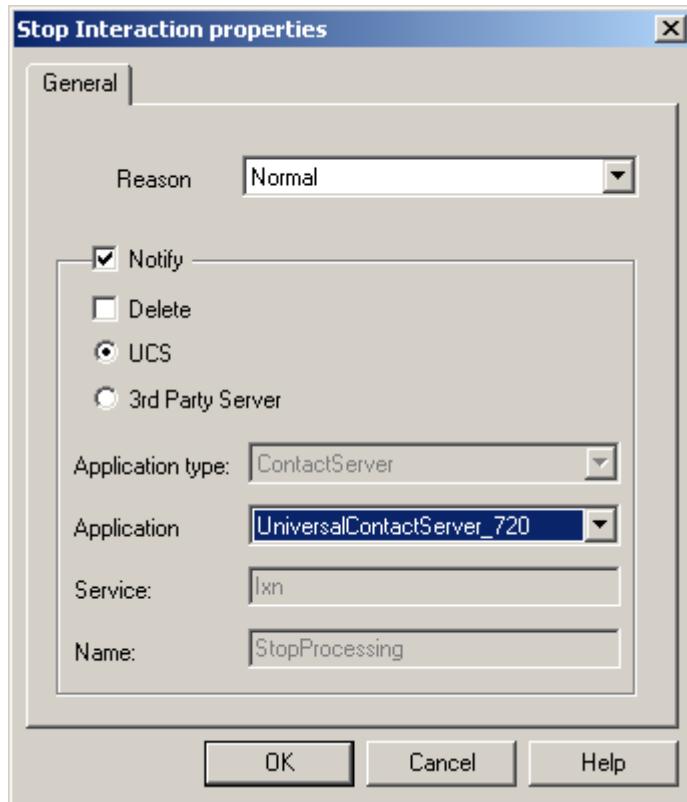


Figure 142: Stop Interaction Properties Dialog Box

Reason Codes

When you place the Stop Interaction object in a strategy, the resulting properties dialog box (see Figure 142) prompts you to select a Reason Code. Configuration Manager predefines the following reason codes in the Business Attributes > StopProcessingReason folder.

- Normal
- AutoResponded
- Terminated
- Sent
- Forwarded
- Redirected

You may wish to extend the list by adding reason codes to the StopProcessReason folder in Configuration Manager that are specific to your site.

Use the information in [Table 78](#) to complete the fields in the General tab.

Table 78: Stop Interaction Object, General Tab

Parameter	Description
Reason	Select a reason code. You may wish to extend the list in Configuration Manager by adding reason codes that are specific to your site. You must select a reason code in order for the strategy to compile. You can also use variables for reason codes.
Notify	Select this check box to enable the Delete, UCS, and Third Party Server fields. In order to update the UCS Database with the Stop reason, the Notify check box must be checked without checking the Delete box.
Delete	Select to have UCS delete the interaction from its database if, for example, the interaction is not required for reporting or history.
UCS	Select to notify UCS that interaction processing has stopped. You can notify either UCS or a third party server; you cannot notify both. If you select this option, you only need to complete the Application type field.
3rd Party Server	If applicable, select the 3rd Party Server radio button and complete the next two fields (the Service and Name fields are optional).
Application type	Click the down arrow and select the Application Type of the third-party server, as it appears in the Configuration database.
Application	Click the down arrow and select the Application Name of the third-party server, as it appears in the Configuration database. Variables can also be used for the Application name.
Service	Optional. Enter the type of notification service for the third-party server. By default, this is Ixn.
Name	Optional. Enter the name of the notification service for the third-party server. By default, this is StopProcessing.

Returned Results for the Stop Interaction Object

The results returned will either be an Event3rdServerResponse message (see [page 151](#)) or an Event3rdServerFault message using the fault codes shown in [Table 79](#):

Table 79: Stop Interaction Object Fault Codes

FaultCode value	FaultString value	Description	Action
201	Missing parameter <parameter_name>	One parameter is missing	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected protocol message from the server's client	End of processing
510	<object_name> not found in database	An object used by the Stop Interaction object is not present in the database	End of processing
701	Unexpected error <error_name>	An unexpected error happened	Retry
710	Connection to <server_name> failed	Connection to the database failed	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry



Submit New Interaction

Use the Submit New Interaction object to define and submit a new interaction to an Interaction Server. This object causes URS to generate a request to the Interaction Server for the method `SubmitNew`.

Submit New Interaction General Tab

[Figure 143](#) shows the General tab in the properties dialog box for the Submit New Interaction object.

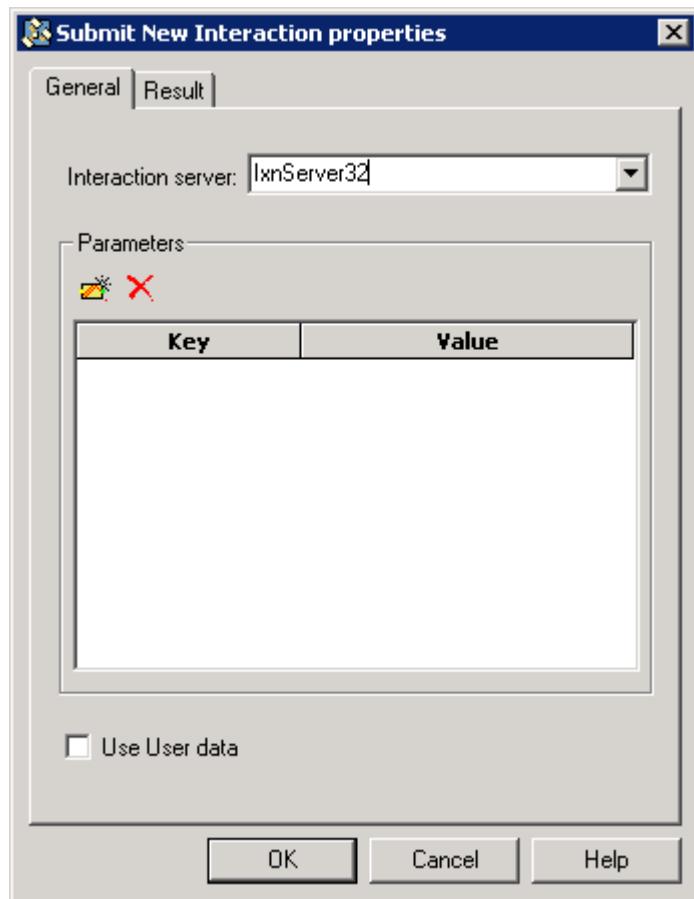


Figure 143: Submit New Interaction Properties Dialog Box

Use the information in [Table 80](#) to complete the General tab of the properties dialog box.

Table 80: Submit New Interaction, General Tab

Parameter	Description
Interaction Server	Select an interaction server from the Interaction server drop-down list, or type the name of the interaction server. Select/Type the server where you want this ESP request to be executed.
Parameters	<p>You can use this pane to specify new interaction properties (which may or may not relate to the current interaction being processed by the strategy). To assign parameter values, click the new item button (). A new row appears under the Key and Value headings. Under Key, property names display for selection. Under Value, specify a new interaction property value. This may be a constant, variable, or expression (including <code>UData</code> function calls).</p> <p>The following system interaction properties are required for all newly submitted interactions (except when the Use User data check box is checked): <code>TenantId</code>, <code>MediaType</code>, <code>InteractionType</code>, <code>InteractionSubtype</code>, and <code>Queue</code>. The following system interaction properties are optional and might or might not be included in the parameters: <code>InteractionId</code>, <code>ParentId</code>, <code>ExternalId</code>, <code>ReceivedAt</code>, <code>IsOnline</code>, <code>InQueues</code>, and <code>OutQueues</code>. Genesys recommends that you do not use the <code>InteractionId</code> parameter or leave it empty, and instead, allow the Interaction Server to generate and return an Interaction ID for the submitted interaction. Also, the user data of the new interaction can be specified with the key appended with a <code>Userdata</code> prefix. For example, if you want to attach a <code>Subject</code> key to the new interaction with value of <code>Your Inquiry</code>, the full key name would be <code>Userdata.Subject</code>, with the value <code>Your Inquiry</code>.</p>
Use User Data	This option was added for those cases in which the exact copy of the current interaction is submitted to a different Interaction Server. For example, for the purpose of archiving and if the current interaction is stopped immediately after the <code>Submit New</code> step. In this case, setting the <code>Parameters</code> option, as in the parameters described above, is not required. Generally, the <code>Use User data</code> check box is left unchecked and all the required <code>Parameters</code> options are specified, unless they are used in a scenario similar to the one described in the row above.

Submit New Interaction Result Tab

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Use the information in Table 20 on [page 173](#) to complete the Result tab in the *Submit New Interaction Properties* dialog box.

Returned Results for the Submit New Interaction Object

The results returned will be either Event3rdServerResponse (see [page 151](#)) or a fault message (Event3rdServerFault, see [page 151](#)).

The content of the Event3rdServerResponse message is shown in [Table 81](#):

Table 81: Submit New Interaction Object Returned Results

Key	Description
InteractionId	This is a newly-generated interaction identifier (or a copy of an input parameter if provided). IRD treats the output value the usual way (save to a variable, attach to user data, or discard).

Any fault message returned uses the fault codes shown in [Table 82](#):

Table 82: Submit New Interaction Object Fault Codes

FaultCode value	FaultString value	Description
105	Invalid request data	The parameters list has not been attached. (This error might not be reproducible.)
108	Invalid date-time format for parameter <parameter_name>	The parameter containing a timestamp is incorrectly formatted
201	Missing parameter name	A required parameter of the request is missing.

Table 82: Submit New Interaction Object Fault Codes (Continued)

FaultCode value	FaultString value	Description
206	Unknown tenant	Unknown tenant ID is specified in the request. (This error might not be reproducible.)
207	Unsupported media type	The specified media type is not supported in this tenant.
208	Unsupported interaction type	The specified interaction type is not supported.
209	Unsupported interaction subtype	The specified interaction subtype is not supported.
210	Unknown queue	The specified queue is not known in this tenant.
211	Invalid interaction userdata	Keys of attached userdata are not formed correctly.
212	Can't set read-only property <property_name>	Attempt to set system read-only property failed
213	Invalid type of <property_name>	A property of invalid type is specified.
215	No database connections available	No database connections are available.
216	Database manager failure	The Database Manager failed.
217	Duplicate interaction identifier	A duplicate interaction ID is specified in the request.



Update Contact

Use the Update Contact object to update contact information in the UCS Database to correspond to the attached values. You can update information for a single contact when ContactId is provided as attached data to the Update Contact object.

You can also use Update Contact to add new fields to the existing contact information in the UCS Database.

Notes: If you want to retrieve one or more existing contacts, use the Identify Contact object (see “Identify Contact” on [page 237](#)).
The ContactId must exist as attached data for the Update Contact object to work.

Update Contact General Tab

[Figure 144](#) shows the properties dialog box for the Update Contact object.

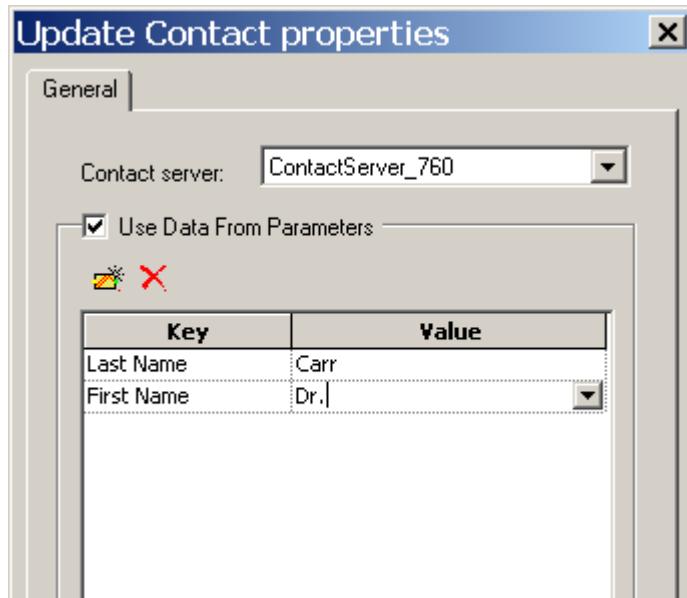


Figure 144: Update Contact Properties Dialog Box

Use the information in [Table 83](#) to complete the General tab of the properties dialog box.

Table 83: Update Contact General Tab

Parameter	Description
Contact Server	Click the down arrow and select a UCS Application from those configured in Configuration Manager.
UseDataFromParameters	By default, UCS updates the contact information based on user data attached to the interaction. If you want to specify different information, select the UseDataFromParameters check box and enter the appropriate values for the parameters selected from the Key drop-down list.

Returned Results for the Update Contact Object

The results returned will be either `Event3rdServerResponse` or `Event3rdServerFault`.

The content of the `Event3rdServerResponse` message is a confirmation that the update proceeded correctly.

Any fault message returned uses the fault codes shown in [Table 84](#):

Table 84: Update Contact Object Fault Codes

FaultCode value	FaultString value	Description	Action
201	Missing parameter <parameter_name>	One parameter is missing	Exit from strategy
203	Incorrect type for parameter <parameter_name>, expected type 1 but was type 2	One parameter has an incorrect type	Exit from strategy
502	Invalid third party message type	Protocol parser encountered an unexpected protocol message from the server's client	End of processing
510	<Object_name> not found in database	An object used by the Stop Interaction object is not present in the database	End of processing
701	Unexpected error <error_name>	An unexpected error happened	Retry
710	Connection to <server_name> failed	Connection to the database failed	Retry
716	Server overloaded, request rejected	The server is overloaded	Retry
732	Invalid Tenant <tenant_name>	The specified tenant name does not exist	Retry

Special Note on the Result Tab in Multimedia Objects

IRD uses the same Result tab for the Acknowledgement ([page 161](#)), Autoresponse ([page 181](#)), Chat Transcript ([page 191](#)), Classify ([page 198](#)), Create Email Out ([page 212](#)), Create Notification ([page 217](#)), Create SMS

([page 221](#)), External Service ([page 115](#)), Identify Contact ([page 237](#)), Screen ([page 263](#)), MultiScreen ([page 241](#)), and Submit New Interaction ([page 277](#)) objects.

Figure 145 shows the Result tab for the Acknowledgement object.

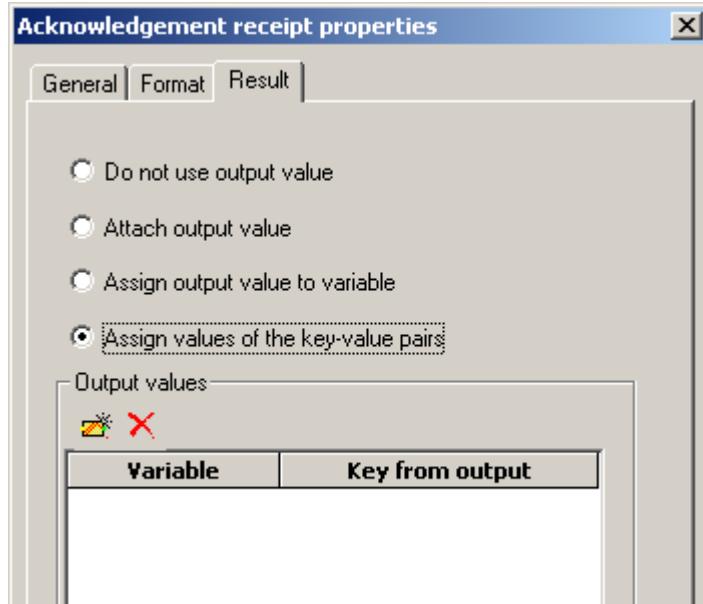


Figure 145: Result Tab

Some options do not apply to all objects. [Table 85](#) lists each of the above objects and indicates when a Result tab option does not apply.

Table 85: Result Tab Options

	A. Do not use output value	B. Attach output value	C. Assign output value to variable	D. Assign values of the key-value pair
Applies to Acknow. Object?	Yes*	Yes**	Yes**	Yes**
Applies to Autoresp. Object?	Yes*	Yes**	Yes**	Yes**
Applies to Chat Trans. Object?	Yes*	Yes**	Yes**	Yes**
Applies to Classify Object?	No	Yes ***	Yes ***	Yes ***
Applies to Create Email Out object?	Yes*	Yes**	Yes**	Yes**

Table 85: Result Tab Options (Continued)

	A. Do not use output value	B. Attach output value	C. Assign output value to variable	D. Assign values of the key-value pair
Applies to Create Notif. object?	Yes*	Yes**	Yes**	Yes**
Applies to Create SMS object?	Yes*	Yes**	Yes**	Yes**
Applies to Create SMS Out object?	Yes*	Yes**	Yes**	Yes**
Applies to Send SMS Out object?	Yes*	Yes**	Yes**	Yes**
Applies to External Service Object?	No	Yes ***	Yes ***	Yes ***
Applies to Identify Contact object?	No	Yes***	Yes***	Yes***
Applies to Screen/MultiScr. Object?	No	Yes ***	Yes ***	Yes ***
Yes* = Further processing of returned data is not always necessary for these objects.				
Yes** = Returned data may be used for reporting or logging purposes.				
Yes*** = A particular mode should be chosen based on required result. If returned data needs to be delivered to agent, use option C. If data is required in a strategy only to make a routing decision, use option C or D.				
No = There is no reason to use an object if not to process returned data.				



Update Interaction

Use the Update Interaction object to update properties of the interaction in the Interaction Server database that are not currently processed in the strategy by URS. This object causes URS to generate a request to Interaction Server for the method `UpdateInteraction`.

The main use case is to update properties of the parent interaction (session or case). This object is used mostly in conjunction with Find Interaction. For example, when you receive a reply onto an outbound e-mail, you can find the parent interaction and change its status to indicate that a reply was received.

Update Interaction General Tab

Figure 146 shows the dialog box that opens when you click the Update Interaction button, place the Update Interaction object in a strategy, and double-click to open its dialog box.

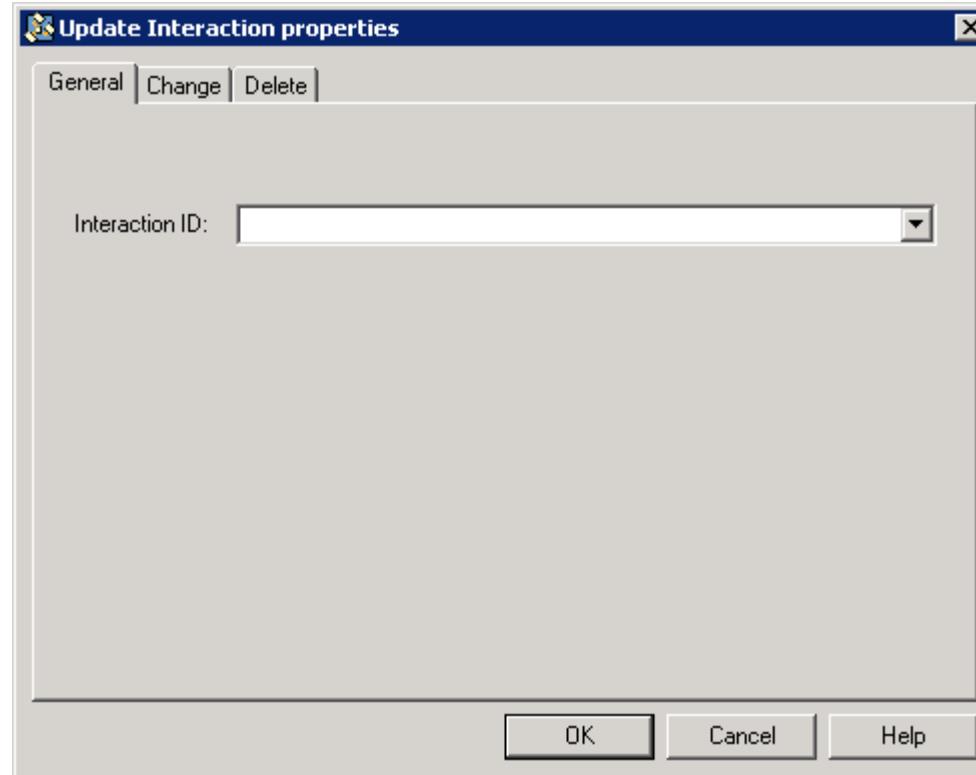


Figure 146: Update Interaction Dialog Box

Use the information in [Table 86](#) to complete the General tab in the Update Interaction Properties dialog box.

Table 86: Update Interaction Object, General Tab

Parameter	Description
Interaction ID	Required parameter that specifies the ID of the interaction to be updated manually or with the strategy variable.

Update Interaction Change Tab

[Figure 147](#) shows the Change tab of the Update Interaction object.

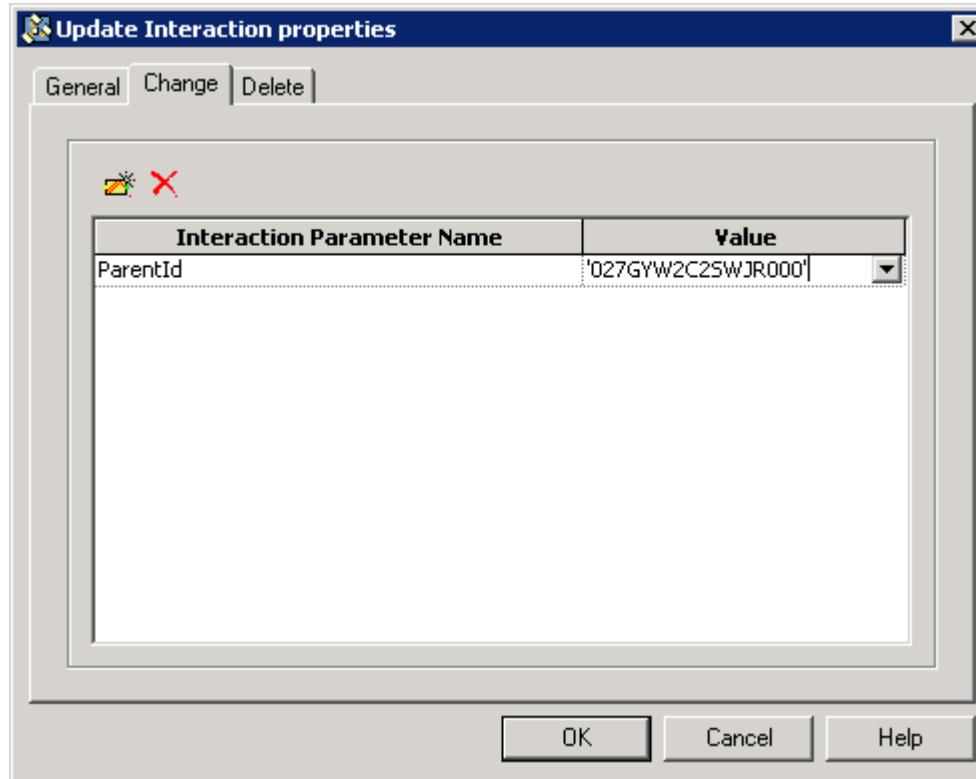


Figure 147: Update Interaction Dialog Box - Change Tab

Note: Any number of parameters can be changed within one request.

Use the information in [Table 87](#) to complete the Change tab in the Update Interaction Properties dialog box.

Table 87: Update Interaction Object, Change Tab

Parameter	Description
Interaction Parameter Name	Specify the name of the interaction parameter to be changed.
Value	Specify a new value for the interaction parameter. The new value can be typed in manually, or you can use a strategy variable instead.

Update Interaction Delete Tab

[Figure 148](#) shows the Delete tab of the Update Interaction object.

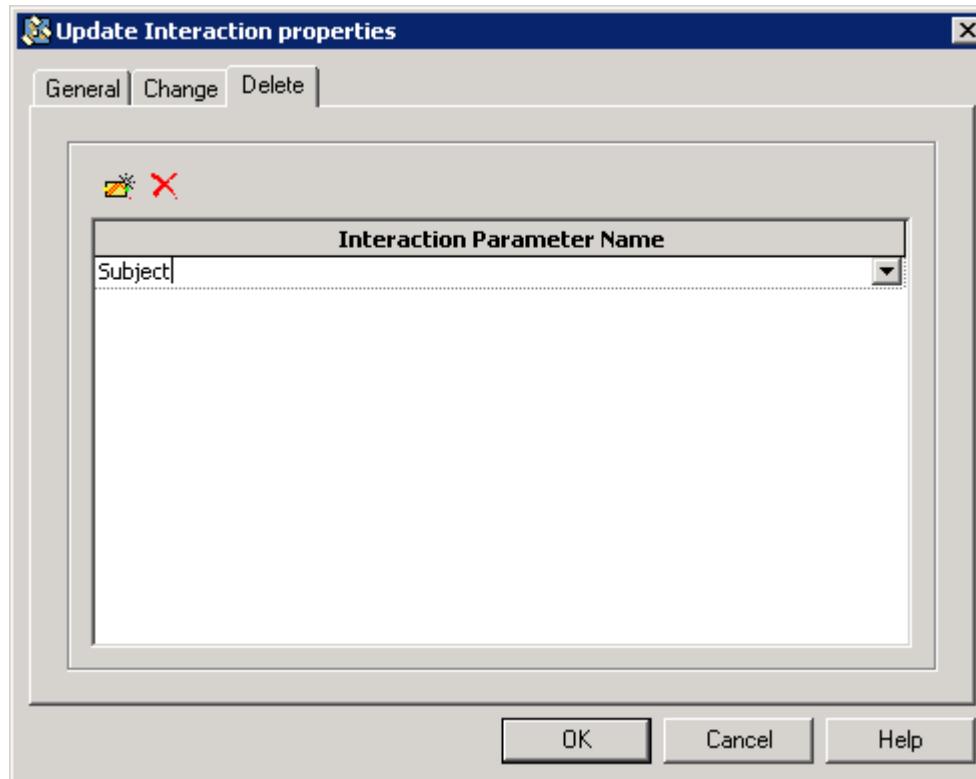


Figure 148: Update Interaction Dialog Box - Delete Tab

Note: Any number of parameters can be deleted within one request.

Use the information in [Table 88](#) to complete the Change tab in the Update Interaction Properties dialog box.

Table 88: Update Interaction Object, Delete Tab

Parameter	Description
Interaction Parameter Name	Specify the name of the interaction parameter to be deleted from interaction properties.

Returned Results for the Update Interaction Object

The results returned will be an Event3rdServerResponse message. The content of the Event3rdServerResponse message is empty.

Any fault message returned uses the fault codes shown in [Table 89](#).

Table 89: Update Interaction Object Fault Codes

FaultCode value	FaultString value	Description	Action
201	Missing parameter 'InteractionId'	The InteractionId parameter is missing	Exit from strategy
206	Unknown tenant	The tenant can not be found	
215	No database connections available	There are no database connections available to carry out the request.	
216	Database manager failure	The Database Manager stopped working or is unresponsive.	
245	Either 'Changed' or 'Deleted' list must be non-empty	Nothing was entered for both the Changed and Deleted parameter lists.	
530	Pull operation failed	The system could not get data from the database.	
536	Tenancy violation	There was an unknown tenancy error.	
537	Interaction manager failure (or database failure)	Either the Interaction Manager or Database Manager failed or became unresponsive.	
543	Unknown interaction identifier specified	The interaction ID that was entered was invalid.	
565	Invalid property type specified	An invalid value was entered for property type.	
566	Attempt to set/change read-only property	The property specified for a Change operation is a read-only property.	
580	Invalid interaction data	The interaction data is invalid or incorrect.	
585	Duplicated properties detected	The system detected duplicated properties and can not continue the operation.	



Update UCS Record

Use the Update UCS Record object to update properties of the interaction in the Universal Contact Server database that are not currently processed in the strategy by URS. This object causes URS to generate a request to Universal Contact Server for the method `UpdateInteraction`.

The main use case is to update properties of the parent interaction (session or case). This object offers a method to propagate changes in an interaction into UCS *manually* at particular selected points in URS interaction processing, since there is currently no immediate synchronization.

Update UCS Record General Tab

Figure 149 shows the dialog box that opens when you click the Update UCS Record button, place the Update UCS Record object in a strategy, and double-click to open its dialog box.

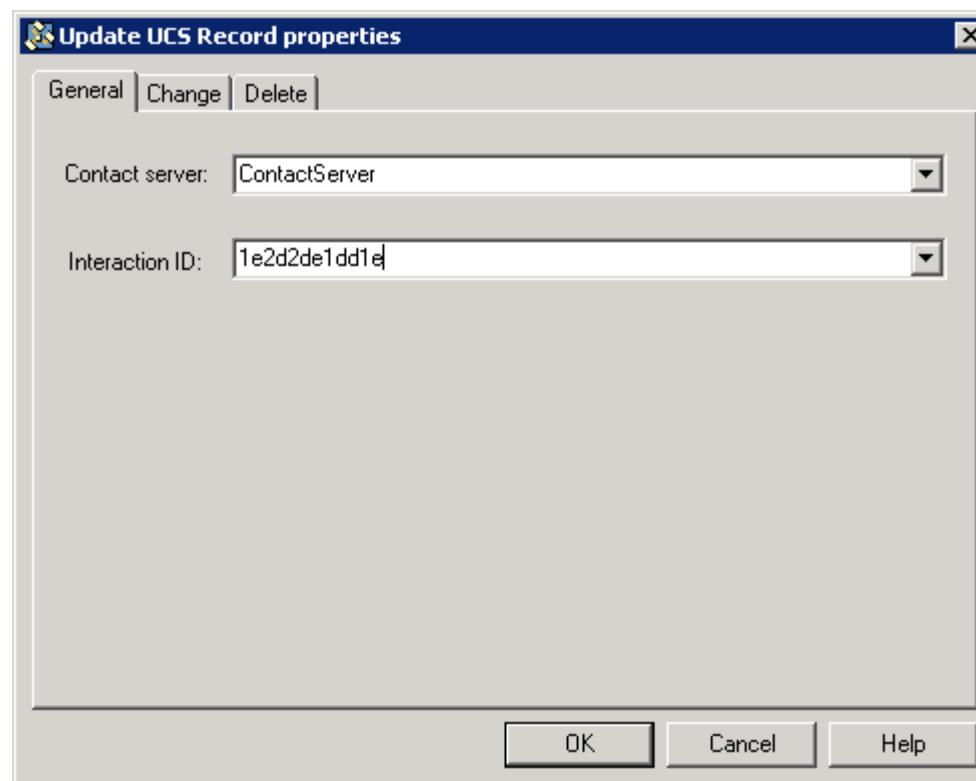


Figure 149: Update UCS Record Dialog Box

Use the information in [Table 90](#) to complete the General tab in the Update Interaction Properties dialog box.

Table 90: Update UCS Record Object, General Tab

Parameter	Description
Contact Server	This is a drop-down list with available Universal Contact Server names.
Interaction ID	Required parameter that specifies the ID of the interaction to be updated manually or with the strategy variable.

Update UCS Record Change Tab

[Figure 150](#) shows the Change tab of the Update UCS Record object.

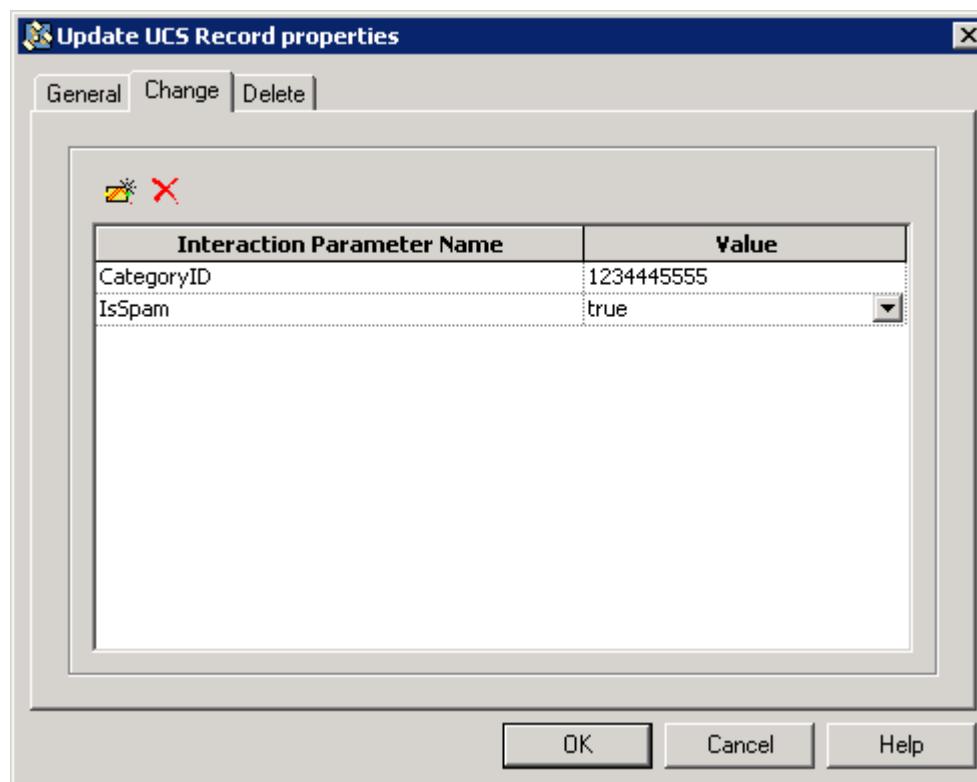


Figure 150: Update UCS Record Dialog Box - Change Tab

Note: Any number of parameters can be changed within one request.

[Table 91](#) shows which interaction and entity attributes can be updated.

Table 91: Interaction and Entity Attributes That Can Be Updated

Interaction Attributes			
TypeId	StartDate	StructuredText	IsSpam
SubtypeId	EndDate	StructTextMimeType	WebSafeEmailStatus
Status	ThreadId	TheComment	IsCategoryApproved
ExternalId	CategoryId	ThreadHash	StoppedReason
OwnerId	Timeshift	CanBeParent	Lang
ContactId	Subject	CreatorAppId	Custom
ParentId	Text	QueueName	
EmailIn Entity Attributes			
FromAddress	ToAddresses	SendDate	EmailOutId
FromPersonal	CcAddresses	Mailbox	ReplyToAddress
BccAddresses	WhichRuleMatched		
EmailOut Entity Attributes			
FromAddress	ToAddresses	SendDate	StandardResponseId
FromPersonal	CcAddresses	ReferenceId	ReplyToAddress
BccAddresses	ReviewerId		
Chat Entity Attributes			
EstablishedDate	ReleaseDate		
PhoneCall Entity Attributes			
Duration	Outcome	PhoneNumber	TConnectionId
Callback Entity Attributes			
CallBackStatus	StartTime	DN	TheType
DetailedDescription	EndTime	Location	Attempts
CustomData	CustomerNumber	CallbackServerID	CallResult
DesiredResponseType			

Use the information in [Table 92](#) to complete the Change tab in the Update Interaction Properties dialog box.

Table 92: Update UCS Record Object, Change Tab

Parameter	Description
Interaction Parameter Name	Specify the name of the interaction parameter to be changed.
Value	Specify a new value for the interaction parameter. The new value can be typed in manually, or you can use a strategy variable instead.

Update UCS Record Delete Tab

[Figure 151](#) shows the Delete tab of the Update UCS Record object.

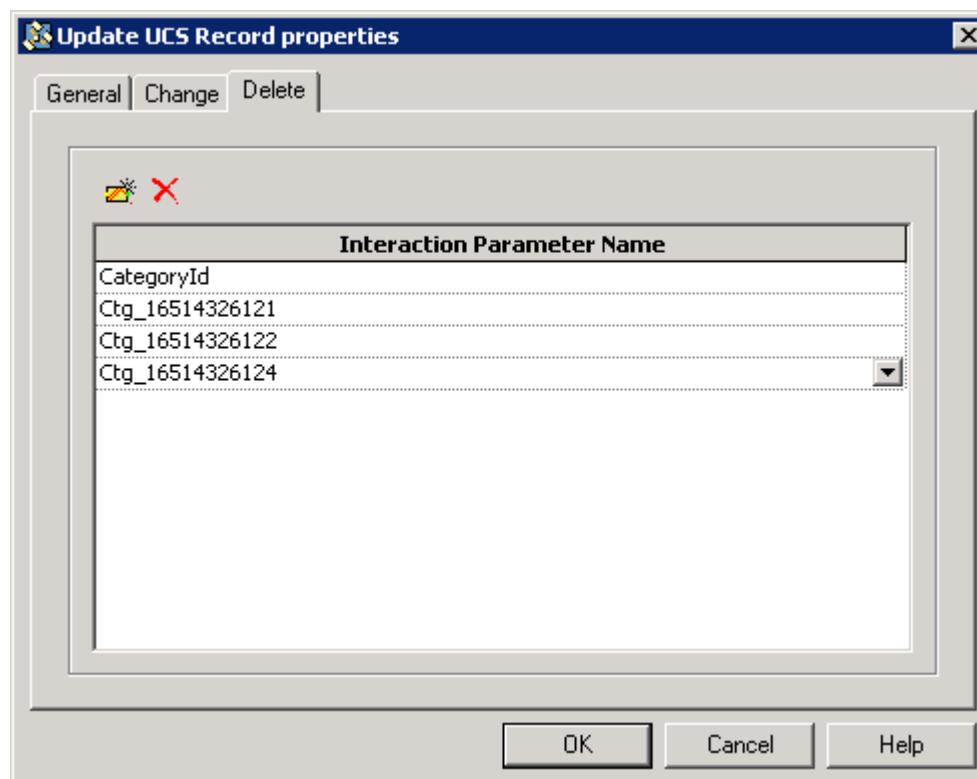


Figure 151: Update UCS Record Dialog Box - Delete Tab

Note: Any number of parameters can be deleted within one request.

Use the information in [Table 93](#) to complete the Delete tab in the Update Interaction Properties dialog box.

Table 93: Update UCS Record Object, Delete Tab

Parameter	Description
Interaction Parameter Name	Specify the name of the interaction parameter to be deleted from interaction properties.

Returned Results for the Update UCS Record Object

The results returned will be an Event3rdServerResponse message. The content of the Event3rdServerResponse message is empty.

Outbound Objects

Note: In order to use the Outbound Objects, Interaction Server must be installed.

The Outbound toolbar in the Routing Design window provides strategy-building objects that support Genesys Outbound Contact, an automated product for creating, modifying, running, and reporting on outbound campaigns for proactive customer contact. [Figure 152](#) shows the buttons for the Outbound objects that appear in the Routing Design window when you click the Outbound icon.

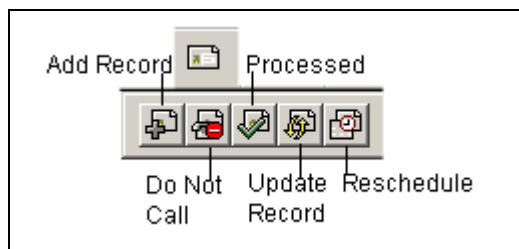


Figure 152: Outbound Objects Toolbar

The Outbound strategy-building objects were specifically designed to be used in strategies configured for proactive routing functionality, the essential element of which is a communication between URS and Interaction Server and Interaction Server and OCS. Thus, all proactive routing interactions will be classified as open-media interactions processed with the `outbound_preview` media type.





Add Record

Use the Add Record object to automate building of *Calling Lists* by adding a new record to a specified Calling List.

Calling List Overview

Calling Lists are database tables with records that store a collection of phone numbers and other customer and business-related data. Each record contains the Campaign name and Campaign ID, as well as unique customer contact information, including the customer's phone number and/or e-mail address. A *Campaign* is a flexible master plan that organizes Calling Lists for the purpose of dialing calls and handling call results.

A Calling List must contain Genesys mandatory fields, such as `contact_info` and `contact_info_type`, and can also contain user-defined, custom fields. The Campaign name, for example, is stored in a user-defined field, which is specified by the `campaign_name_field` option. A user-defined field may also serve as a customer identifier for Do Not Call requests, as an alternative to the DNC restriction on a customer's phone number.

Use Case

Use Add Record to automatically develop a Calling List, such as one to follow up on inbound calls that were abandoned during traffic peaks. You can then configure a routing strategy to detect abandoned calls and add records to the Calling List with the parameters of the incoming interactions. The Calling List can subsequently be used by an outbound campaign that dials out these customers during off peak hours and has the agent apologize and follow up.

Add Record Object Properties

Figure 153 on [page 296](#) shows the Add Record Properties dialog box.

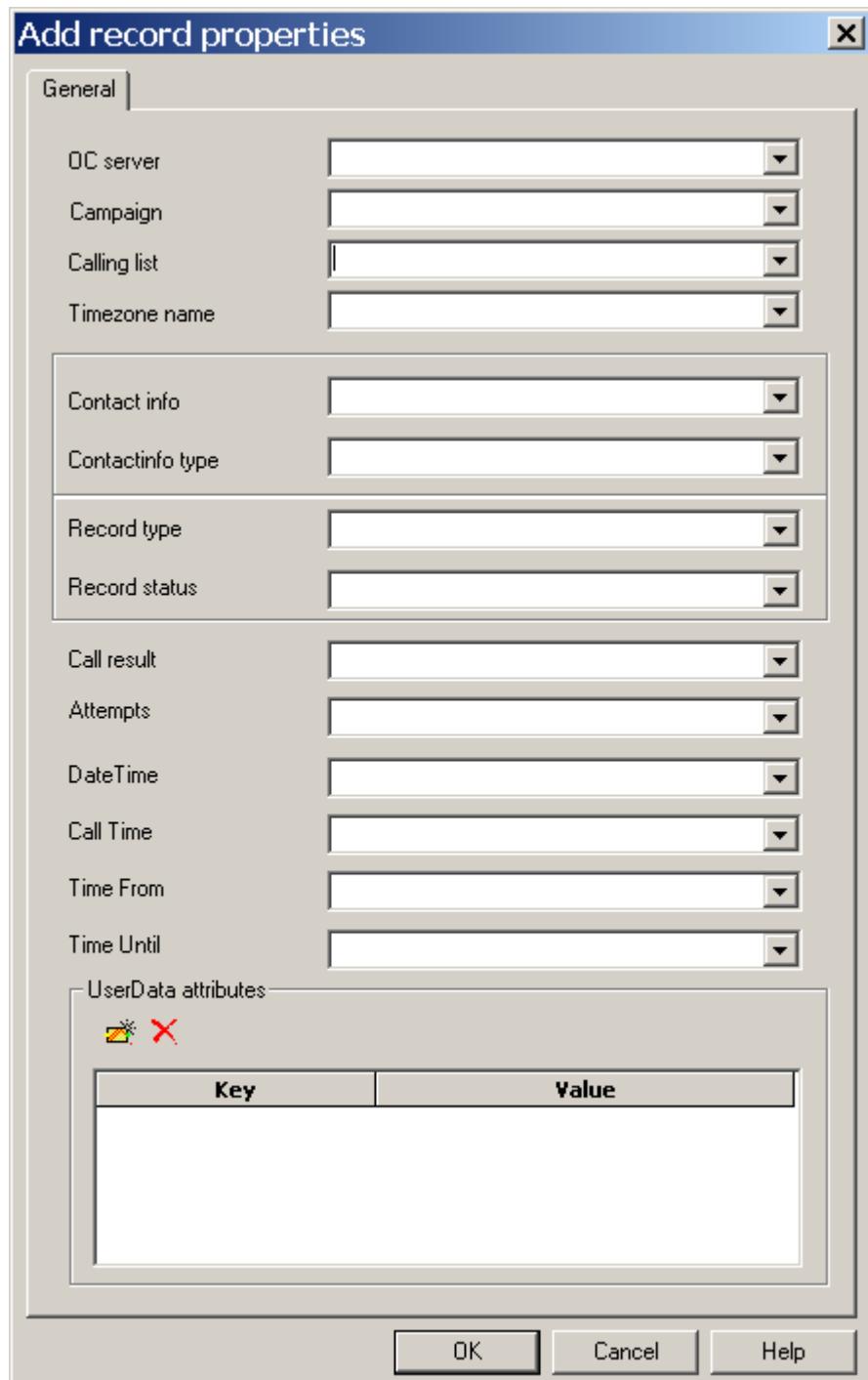


Figure 153: Add Record Properties Dialog Box

Use the information in [Table 94](#) to complete the dialog box. If you require more information on adding records to Calling Lists in general, consult the section on adding records to a Calling List in the *Outbound Contact 8.1 Reference Manual*.

Table 94: Add Record General Tab

Parameter	Description
OC server	Select an Outbound Contact Server that URS will request (through Interaction Server) to create the Calling List record. If not specified, Interaction Server will use the first Outbound Contact Server in its Connections List.
Campaign	Required field. Enter the variable or select the Campaign name to be run. Run time specification allowed.
Calling list	Required field. Select the name (or variable containing the name) of the Calling List for the current interaction (see “Calling List Overview” on page 295). Run time specification allowed.
Timezone Name	Required field. Select a Configuration Server Time Zone name to associate with this record. Run time specification allowed.
Contact info	Required field. Enter or select a phone number or e-mail address for the customer.
Contactinfo type	Optional field. Select the customer’s contact type, phone or e-mail, as defined in a Configuration Manager Enumeration table, or leave empty.
Record Type	Optional field. Select the record type code, as defined in a Configuration Manager Enumeration table, or leave empty.
Record Status	Optional field. Select a record status code, as defined in a Configuration Manager Enumeration table, or leave empty.
Call Result	Optional field. Select a result code, as defined in a Configuration Manager Enumeration table, or leave empty.
Attempts	Optional field. Select/enter the number of attempts to create the record or leave empty and use the default of 0. Run time specification allowed.
DateTime	Optional field. Select/enter when to schedule the callback or leave empty to use the default of 0. Use format MM/DD/YY(YYYY) HH:MM. Run time specification allowed.
Call Time	Optional field. Select/enter the system time when the record was called, in seconds from 01/01/1970 (GMT) 00:00 or leave empty. Run time specification allowed.

Table 94: Add Record General Tab (Continued)

Parameter	Description
Time From	Optional field. Select/enter the “from” portion of the timeframe when the record can be called. Use seconds from midnight or leave empty to use the default of 28800 (8:00 AM). Run time specification allowed.
Time Until	Optional field. Select/enter the “until” portion of the time frame when the record can be called. Use seconds from midnight or leave empty to use the default of 64800 (6:00 PM). Run time specification allowed.
UserData attributes	Optional field. Use this area to define key-value pairs to be added/updated in the User Data. To define key-value pairs, click the new item button (). A new row appears under the Key and Value headings. Under Key in the new row, select an existing User Data key. Under Value, select a strategy variable that contains the value or enter a value.

Note: Due to the different sources in the Configuration Database, the Call Result values list provided by the Add Record object can be inconsistent with the list of Call Result values populated in a record in Outbound Contact Manager.

Returned Results for the Add Record Object

The results returned will be either Event3rdServerResponse (see [page 151](#)) or a fault message (Event3rdServerFault, see [page 151](#)).

The content of the Event3rdServerResponse message for the AddRecord method is shown in [Table 95](#):

Table 95: Add Record Method Returned Results

FaultCode value	FaultString value	Description
101	No Active Campaigns	Can not execute request. Calling list was not loaded.
102	No Running Preview Campaigns	Not applicable.
103	No Records Available	Not applicable.
104	Record not found	Outbound Contact Server received request regarding not existing or already processed record.
105	Invalid Request Data	Some mandatory parameters are missed.

Table 95: Add Record Method Returned Results (Continued)

FaultCode value	FaultString value	Description
106	Invalid Request	Invalid method.
107	Invalid Time	Received time does not meet request conditions (e.g., reschedule in the past).
108	Invalid Time Format	Outbound Contact Server cannot convert string to time (e.g., “25/45/00”).
109	Record Already Processed	Received request refers to record already processed.
110	DB Error	Cannot execute the request due to database error.
111	Chained Records not found	Not applicable.
112	Record already exists	Attempt to add record existing in database.
113	AddRecord Error	Cannot add the record.
114	Scheduled record not found	Not applicable.



Do Not Call

Use the Do Not Call object to add a phone number or e-mail address to a specified Do Not Call List and mark the corresponding record as Do Not Call.

Note: Use the Do Not Call and Processed (page 302) objects to finalize Outbound record processing. You cannot use other Outbound objects to process records with the same Record Handle after using Processed or DoNotCall in a strategy flow.

Do Not Call List Overview

In the Outbound Contact product, a list of customers who request not to be contacted is known as a Do Not Call List. This data is stored in a Do Not Call List file, using as the key either the customer's contact information, such as a phone number and e-mail address, or a customer ID.

Outbound Contact Manager processes Do Not Call Lists that are formatted as plain text (flat) files, with columns of data separated by commas (comma-separated values), and rows separated by the end of a line.

Use the Do Not Call object to prevent a record from being dialed by any campaign. If a record is marked as Do Not Call, Outbound Contact Manager rejects all subsequent requests for the record.

When URS executes this object in a strategy, it sends (via Interaction Server) a DoNotCall request to OCS. Using the GSW_RECORD_HANDLE provided, OCS:

- Identifies the record and updates the record type as NoCall.
- Enters the phone number and e-mail address or customer ID of this record in the gsw_donotcall_list (table).

If you require more information on Do Not Call Lists in general, see the section on submitting Do Not Call requests in the *Outbound Contact 8.1 Reference Manual*.

Do Not Call Object Properties

Figure 154 shows the Do Not Call Properties dialog box.

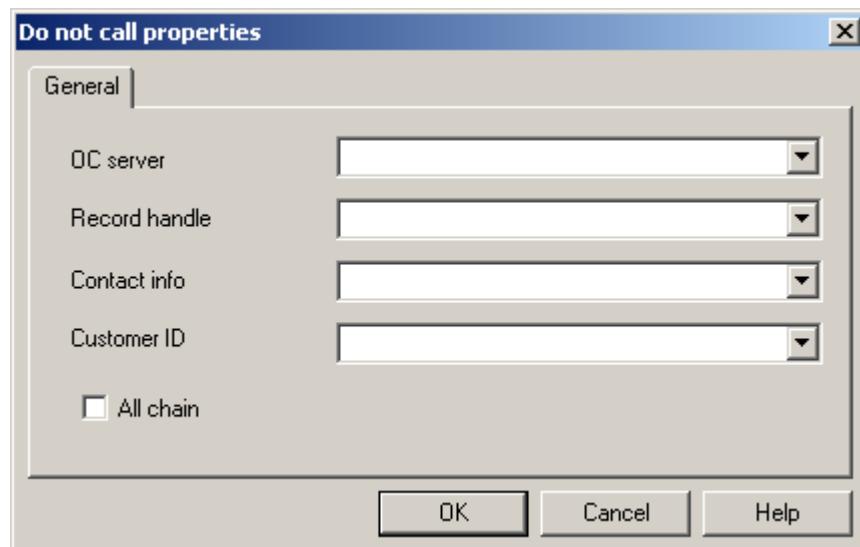


Figure 154: Do Not Call Properties Dialog Box

Use the information in Table 96 to complete the dialog box.

Table 96: Do Not Call General Tab

Parameter	Description
OC server	Select an Outbound Contact Server that URS will request (through Interaction Server) to create the Do Not Call List record. If not specified, Interaction Server will use the first Outbound Contact Server in its Connections List.
	Specify one of the following: <ul style="list-style-type: none"> • Record handle • Contact info • Customer ID
RecordHandle	Enter or select a variable for the Record ID assigned by Outbound Contact Server. Run time specification allowed.
ContactInfo	Enter or select a variable for the phone number or e-mail address. Run time specification allowed.
CustomerID	Enter or select a variable for the CustomerID. Run time specification allowed.
AllChain	This check box contains a flag (Check or Uncheck) determining whether to update the record chain or just the single record. <ul style="list-style-type: none"> • Uncheck: Only one phone number will be added to the Do Not Call List. • Check if you want to add all phone numbers from the record chain.

Returned Results for Do Not Call Object

The results returned will be either Event3rdServerResponse (see [page 151](#)) or a fault message (Event3rdServerFault, see [page 151](#)).

The content of the Event3rdServerResponse message for the Do Not Call method is shown in [Table 97](#).

Table 97: Do Not Call Method Returned Results

FaultCode Value	FaultString Value	Description
101	No Active Campaigns	Cannot execute request. Campaign was not loaded.
102	No Running Preview Campaigns	Not applicable.
103	No Records Available	Not applicable.

Table 97: Do Not Call Method Returned Results (Continued)

FaultCode Value	FaultString Value	Description
104	Record not found	OCS received request regarding non-existing or already processed record.
105	Invalid Request Data	Some mandatory parameters are missed.
106	Invalid Request	Invalid method.
107	Invalid Time	Not applicable.
108	Invalid Time Format	Not applicable.
109	Record Already Processed	Received request refers to record already processed.
110	DB Error	Cannot execute the request due to database error.
111	Chained Records not found	Not applicable.
112	Record already exists	Attempt to add record existing in database.
113	AddRecord Error	Not applicable.
114	Scheduled record not found	Not applicable.



Processed

Note: Use the Processed and Do Not Call (page 299) objects to finalize Outbound record processing. You cannot use other Outbound objects to process records with the same Record Handle after using Processed or DoNotCall in a strategy flow.

In the Outbound Contact product, when an agent finishes processing a Calling List record (see [page 299](#)), Genesys Desktop sends a RecordProcessed event to indicate that the record is processed and OCS updates the record accordingly. Use the Processed object in a strategy to have URS request (through Interaction Server) that Outbound Contact Server finish processing a record created as a result of a customer call that was previously:

- Initiated from a Calling List (see “Calling List Overview” on [page 295](#))
- Automatically dialed by the Genesys Outbound Contact product
- Routed to an agent when the potential customer answered

When URS executes this object in a strategy, it results in an External Service Request (through Interaction Server) to Outbound Contact Server. Since the

request goes through Interaction Server, you must have that component installed.

Working with User Data

URS can access Proactive record data as User Data in Interaction Server/T-Server event messages (`EventRouteRequest` in most cases). You can use the standard User Data access functions `UData[key]`, `InteractionData[key]`, `BusinessData[key]` to get the data. Campaign identification is available as User Data with key `GSW_CAMPAIGN_NAME`, Calling List as User Data with key `GSW_CALLING_LIST`, record identifier as `GSW_RECORD_HANDLE`, customer e-mail address or phone as `GSW_PHONE`.

To manipulate attached data, use function `Attach[key, value]` to add user-defined data ([page 451](#)), function `Update[key, value]` to update ([page 467](#)), and `DeleteAttachedData[key]` to delete ([page 456](#)).

For more information, see the section on updating call results, as well as Genesys events and event type protocols, in the *Outbound Contact 8.1 Reference Manual*.

Processed Object Properties

[Figure 155](#) shows the Processed Properties dialog box.

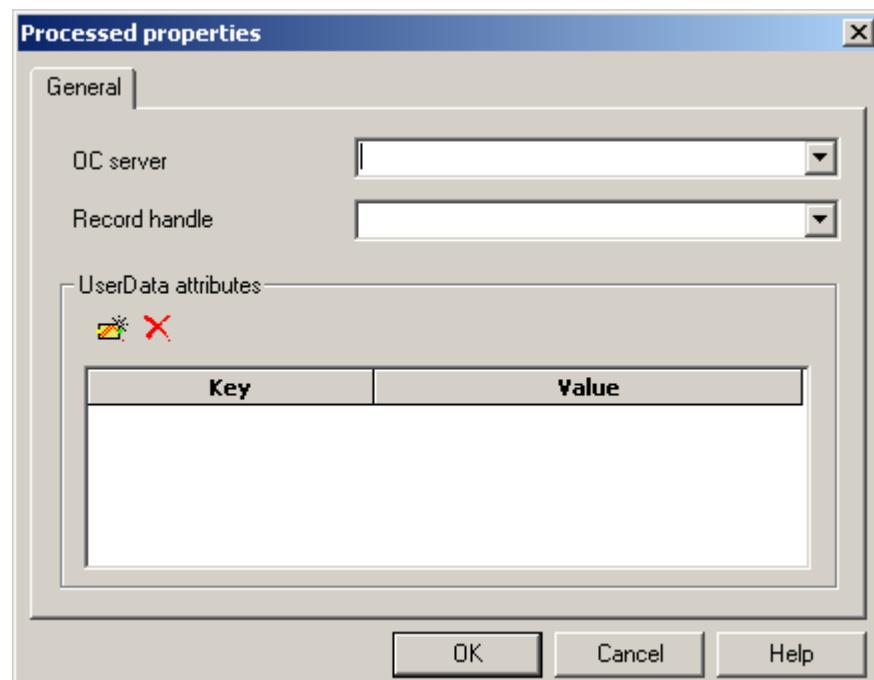


Figure 155: Processed Properties Dialog Box

Use the information in [Table 98](#) to complete the dialog box.

Table 98: Processed General Tab

Parameter	Description
OC server	Select an Outbound Contact Server that URS will request (through Interaction Server) to finish processing the Calling List record. If not specified, Interaction Server will use the first Outbound Contact Server in its Connections List.
RecordHandle	Required field. Enter or select a variable for the Record ID assigned by Outbound Contact Server. Run time specification allowed.
UserData attributes	<p>Use this area to define key-value pairs to be added/updated in the User Data.</p> <p>To define key-value pairs, click the new item button (). A new row appears under the Key and Value headings. Under Key in the new row, select an existing User Data key. Under Value, select a strategy variable that contains the value or enter a value. Run time specification allowed for values, not keys.</p>

Returned Results for Processed Object

The results returned will be either Event3rdServerResponse (see [page 151](#)) or a fault message (Event3rdServerFault, see [page 151](#)).

The content of the Event3rdServerResponse message for the Processed method is shown in [Table 99](#):

Table 99: Processed Method Returned Results

FaultCode Value	FaultString value	Description
101	No Active Campaigns	Can not execute request. Campaign was not loaded.
102	No Running Preview Campaigns	Not applicable.
103	No Records Available	Not applicable.
104	Record not found	OCS received request regarding non-existing or already processed record.
105	Invalid Request Data	Some mandatory parameters are missed.
106	Invalid Request	Invalid method.
107	Invalid Time	Received time does not meet request conditions (e.g., reschedule in the past).

Table 99: Processed Method Returned Results (Continued)

FaultCode Value	FaultString value	Description
108	Invalid Time Format	OCS cannot convert string to time (e.g., “25/45/00”).
109	Record Already Processed	Received request refers to record already processed.
110	DB Error	Cannot execute the request due to database error.
111	Chained Records not found	Not applicable.
112	Record already exists	Attempt to add record existing in database.
113	AddRecord Error	Cannot add the record.
114	Scheduled record not found	Not applicable.



Reschedule

Use the Reschedule object to reschedule a customer call on a Calling List (see “Calling List Overview” on [page 295](#)). A record is typically rescheduled during a call when a customer requests a callback at a certain time.

As described in the section on scheduling and rescheduling records in the *Outbound Contact 8.1 Reference Manual*, Outbound Contact supports two methods for rescheduling records:

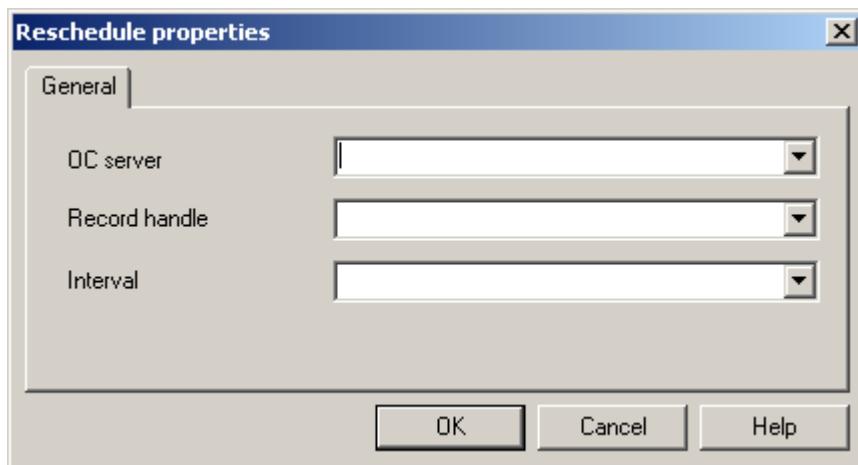
1. Using RecordReschedule to reschedule a call.
2. Using ScheduledRecordReschedule when a rescheduled call cannot be completed and must be set for another time.

The Reschedule object emulates the first method. URS sends OCS (through Interaction Server) a RecordReschedule message and receives a RecordRescheduleAcknowledge in return.

When URS executes this object in a strategy, it results in an External Service Request (through Interaction Server) to Outbound Contact Server. Since the request goes through Interaction Server, you must have that component installed.

Reschedule Object Properties

[Figure 156](#) shows the General tab in the Reschedule Properties dialog box.

**Figure 156: Reschedule Properties Dialog Box**

Use the information in [Table 100](#) to complete the dialog box.

Table 100: Reschedule General Tab

Parameter	Description
OC server	Select an Outbound Contact Server that URS will request (through Interaction Server) to reschedule the Calling List record. If not specified, Interaction Server will use the first Outbound Contact Server in its Connections List.
RecordHandle	Required field. Enter or select a variable for the Record ID assigned by Outbound Contact Server. Run time specification allowed.
Interval	Required field. In order to schedule the requested callback based on the customer's requested callback time, enter or select a variable for the interval in minutes from the requested callback time. After this interval, the Calling List record will be re-submitted to Interaction Server for submittal to Outbound Contact Server for rescheduling. Run time specification allowed.

Returned Results for Reschedule Object

The results returned will be either Event3rdServerResponse (see [page 151](#)) or a fault message (Event3rdServerFault, see [page 151](#)).

The content of the Event3rdServerResponse message for the Reschedule method is shown in [Table 101](#):

Table 101: Reschedule Method Returned Results

FaultCode Value	FaultString Value	Description
101	No Active Campaigns	Cannot execute request. Campaign was not loaded.
102	No Running Preview Campaigns	Not applicable.
103	No Records Available	Not applicable.
104	Record not found	OCS received request regarding non-existing or already processed record.
105	Invalid Request Data	Some mandatory parameters are missed.
106	Invalid Request	Invalid method.
107	Invalid Time	Received time does not meet request conditions (e.g., reschedule in the past).
108	Invalid Time Format	OCS cannot convert string to time (e.g., “25/45/00”)
109	Record Already Processed	Received request refers to record already processed.
110	DB Error	Cannot execute the request due to database error.
111	Chained Records not found	Not applicable.
112	Record already exists	Attempt to add record existing in database.
113	AddRecord Error	Cannot add the record.
114	Scheduled record not found	Not applicable.



Update Record

Use this object to update a Calling List record (see “Calling List Overview” on [page 295](#)) that you specify via a RecordHandle parameter.

The `UpdateCallCompletionStats` request updates Genesys modifiable mandatory fields and custom fields in a record to OCS. In Predictive dialing mode, this request can be used to overwrite the call result detected by call progress detection when needed. Or you can overwrite a call result `answer` with the call result `wrong party`.

When URS executes this object in a strategy, it results in an External Service Request (through Interaction Server) to Outbound Contact Server. Since the request goes through Interaction Server, you must have that component installed.

For more information on updating records, see the section on updating call results and custom fields in the *Outbound Contact 8.1 Reference Manual*.

Processed Object Properties

[Figure 157](#) shows the Update Properties dialog box.

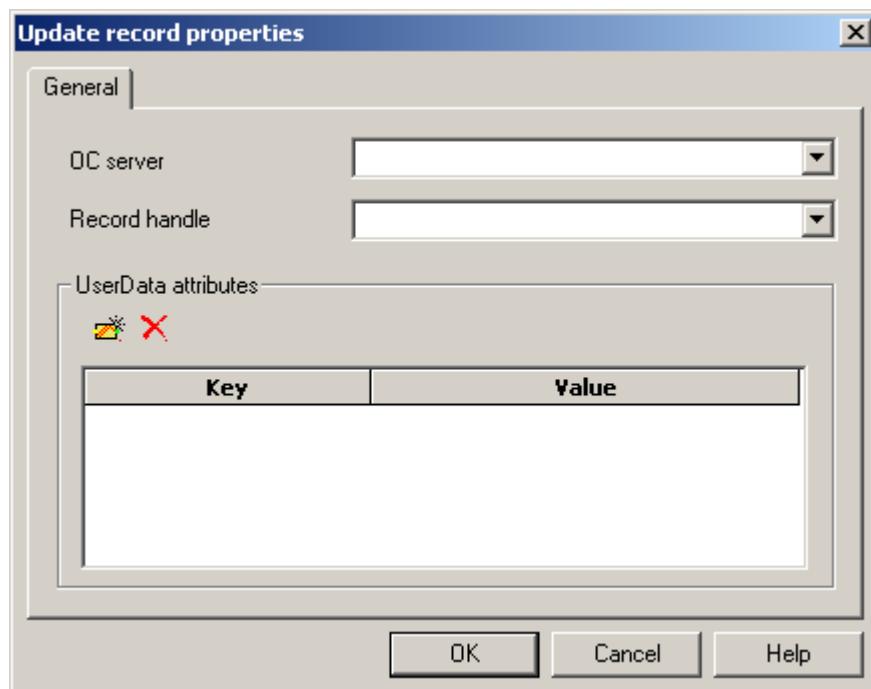


Figure 157: Update Record Properties Dialog Box

Use the information in [Table 102](#) to complete the dialog box.

Table 102: Update Record Object, General Tab

Parameter	Description
OC server	Select an Outbound Contact Server that URS will request (through Interaction Server) to update the Calling List record. If not specified, Interaction Server will use the first Outbound Contact Server in its Connections List.

Table 102: Update Record Object, General Tab (Continued)

Parameter	Description
Record handle	Enter or select a variable for the Record ID assigned by Outbound Contact Server. Run time specification allowed.
UserData attributes	<p>Use this area to define key-value pairs to be added/updated in the User Data.</p> <p>To define key-value pairs, click the new item button (). A new row appears under the Key and Value headings. Under Key in the new row, select an existing User Data key. Under Value, select a strategy variable that contains the value or enter a value. Run time specification allowed for values, not for keys.</p>

Returned Results for Reschedule Object

The results returned will be either Event3rdServerResponse (see [page 151](#)) or a fault message (Event3rdServerFault, see [page 151](#)).

The content of the Event3rdServerResponse message for the Reschedule method is shown in [Table 103](#):

Table 103: Reschedule Method Returned Results

FaultCode Value	FaultString Value	Description
101	No Active Campaigns	Can not execute request - no one campaign was loaded
102	No Running Preview Campaigns	Not applicable
103	No Records Available	Not applicable
104	Record not found	OCS received request regarding not existing or already processed record
105	Invalid Request Data	Some mandatory parameters are missed
106	Invalid Request	Invalid method
107	Invalid Time	Received time does not meet request conditions (e.g. reschedule in the past)
108	Invalid Time Format	OCS cannot convert string to time (e.g. "25/45/00")
109	Record Already Processed	Received request refers to record already processed
110	DB Error	Cannot execute the request due to DB error

Table 103: Reschedule Method Returned Results (Continued)

FaultCode Value	FaultString Value	Description
111	Chained Records not found	Not applicable
112	Record already exists	Attempt to add record existing in DB
113	AddRecord Error	Cannot add the record
114	Scheduled record not found	Not applicable

Configuring Proactive Routing

Notes: For information on enabling proactive interaction functionality on the agent desktop (also known as Push Preview mode), see the chapter on Communication Protocols, Proactive Interaction Support section, in the *Outbound Contact 8.1 Reference Manual*.

Also see the *Genesys 7.6 Proactive Routing Solution Guide*.

Keep the following in mind when configuring proactive routing:

- You can configure routing strategies to distinguish proactive interactions (outbound records) from inbound interactions.

Preview records are submitted with a Media Type (see Figure 48 on [page 102](#)) of `outbound_preview`. Outbound Contact Server (OCS) delivers them according to Agent Capacity rules and current agent activity. Agents may process them in parallel with any other kind of interactions.

To be able to receive preview records, the agent must be logged into an Interaction Server Place and into Media Channel `outbound_preview`. To access an interaction's Media Type, use functions `GetMediaTypeName[mediaId]` and `GetMediaType[]`. If `(GetMediaTypeName[GetMediaType[]]='outbound_preview')` then ... else ...

Table 104: OCS Media Type Specification

Key	Type	Value
<code>GSW_CONTACT_MEDIA_TYPE</code>	String	Media type corresponding to Media Type Business Attribute value. Use standard user data access functions such as <code>UData[key]</code> to access user data including <code>GSW_CONTACT_MEDIA_TYPE</code> .

- Universal Routing is not hardcoded to handle rejected interactions. Similar to Ring-No-Answer situations, you must configure routing strategies to handle rejected interactions such as:

Restoring rejected interaction positions in waiting queues by using age of interaction.

- Marking rejected interactions with attached data then analyze the attached data.

Implementation Sample

Strategy that routes to agents step-marks interactions before routing with attached data, `LastStepRoutingPoint`, and the value of `Dest[]` function (returns value of `ThisDN` in event message).

Rejected interactions are resubmitted to the same strategy on the same routing point.

The same strategy should at the start compare the `LastStepRoutingPoint` attached data with the value of the `Dest[]` function. If the same, it is very likely the interaction was rejected so some specific steps it should be done here.

Using age of interaction makes it possible to place rejected interactions in waiting queues in the correct order. If necessary, the strategy can also check the name of the agent to call was routed to for the rejected interaction in order to exclude this agent (`ExcludeAgents[]` function) as a potential routing target.

Routing Objects

Note: Use Routing objects with corresponding routing rules, as well as Selection object, for voice interactions only. Use the remaining three Routing objects (Route Interaction, Workbin and Queue Interaction) for non-voice interactions only.

Figure 158 shows the buttons on the Routing objects toolbar that opens when you click the Routing icon.



Figure 158: Icon for Routing Objects

IRD can select a target using Routing objects. The Service Level, Load Balancing, Percentage, Statistics, Switch to Strategy, Force Routing, and Workforce objects select the targets based on user-defined routing rules, which can be used in more than one strategy.

Creating Routing Rules for Routing Objects

Before using a Routing object that uses a routing rule (see [page 70](#)), if the applicable routing rule does not exist, you must create it. Open the **Routing Rules** list in IRD and then select **New** from the **File** menu. This opens the **New Routing Rules** dialog box (see [Figure 159](#)).

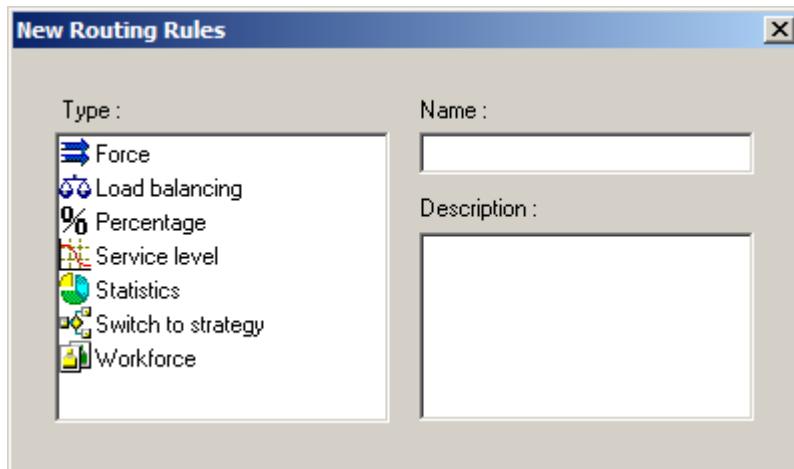


Figure 159: New Routing Rules Dialog Box

After you select the routing rule type and enter a name and description, a routing rule dialog box opens.

Properties Dialog Boxes for Routing Rules

Figure 160 on [page 313](#) shows the **Percentage** tab of a completed **Percentage routing rule** dialog box.

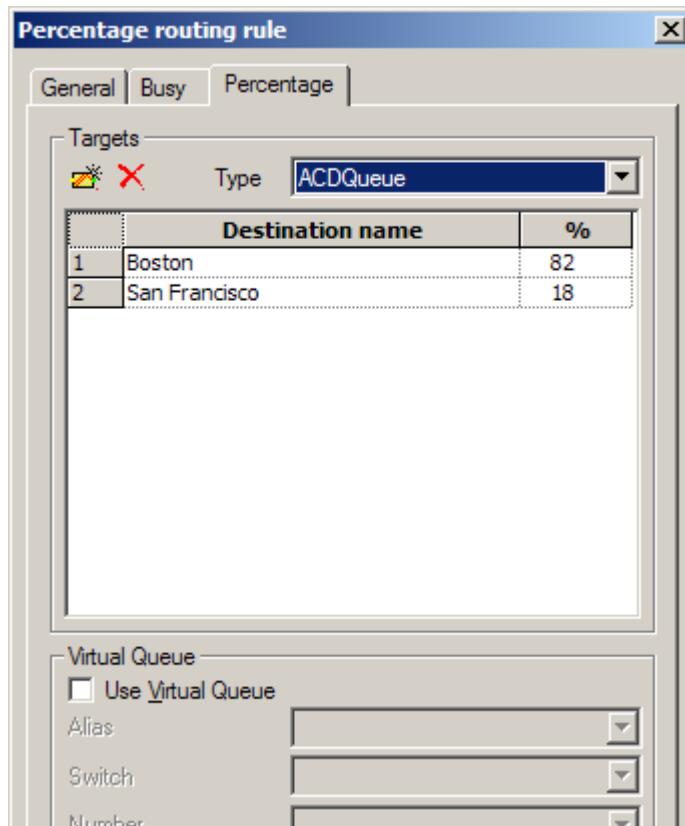


Figure 160: Percentage Routing Rule Dialog Box

Figure 161 shows the Load balancing tab of a completed Load balance routing rule dialog box.

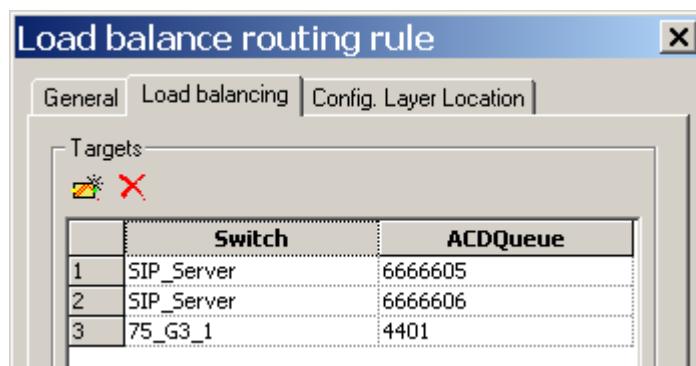


Figure 161: Load Balance Routing Rule Dialog Box

Note: For more information on loading balancing, see the respective chapter in the *Universal Routing 8.1 Deployment Guide*.

Figure 162 on [page 314](#) shows the Statistics tab for an example Statistics routing rule.

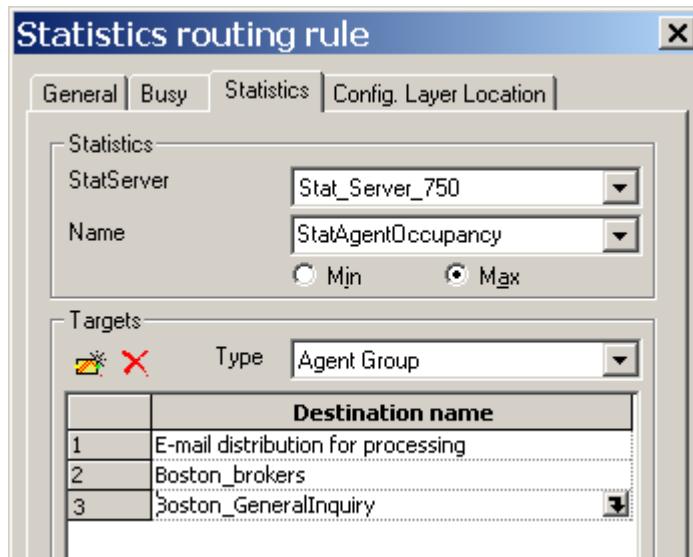


Figure 162: Statistics Routing Rule Dialog Box

Figure 163 shows the Service Level tab for an example Service Level routing rule.

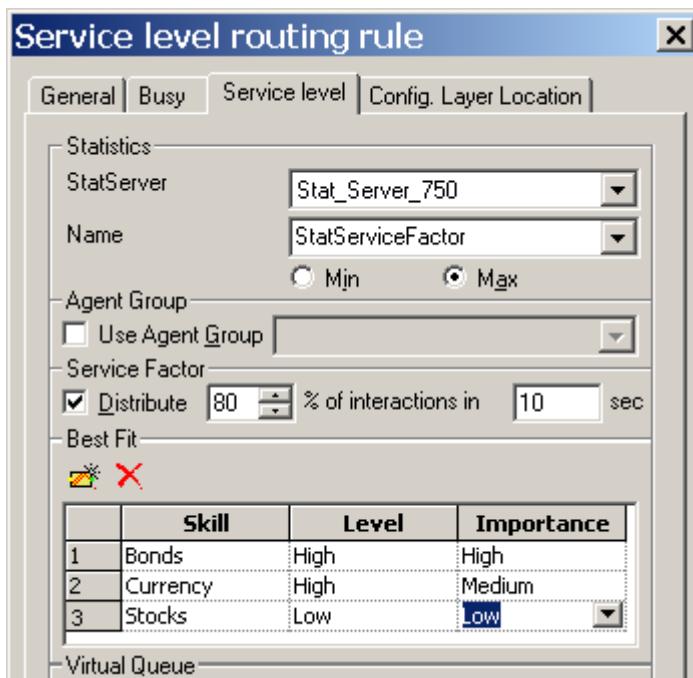


Figure 163: Service Level Routing Rule Dialog Box

Figure 164 on [page 315](#) shows the Workforce tab for an example Workforce routing rule.

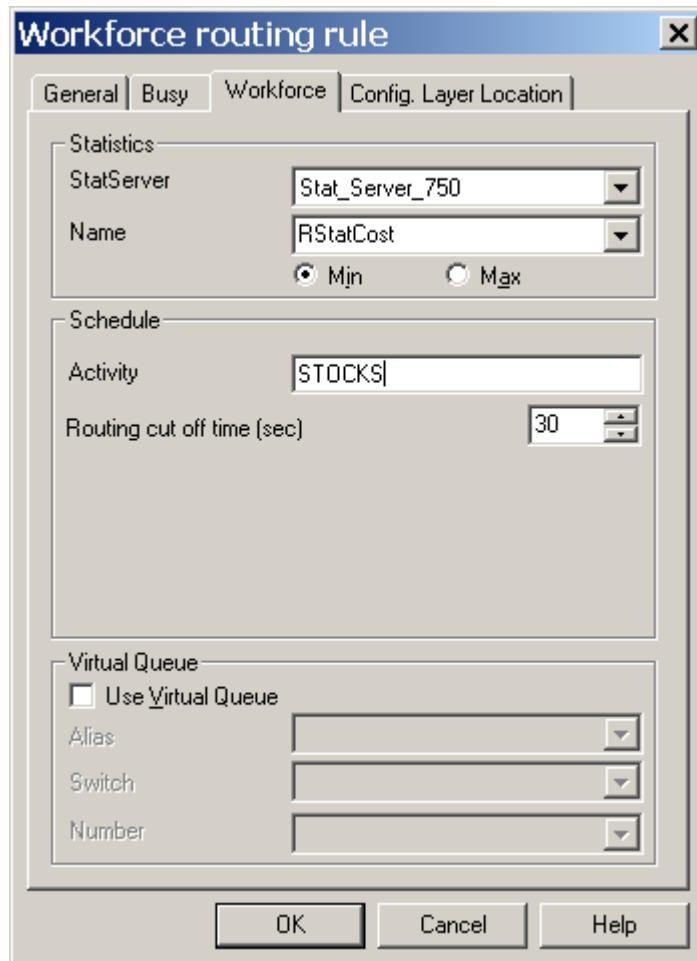


Figure 164: Workforce Routing Rule Dialog Box

- For an example Force routing rule, see Figure 165 on [page 316](#).
- For an example Switch-to-Strategy routing rule, see Figure 178 on [page 347](#).

Once you have created a routing rule, information about it appears when it is selected in the properties dialog box for the selected Routing object (see Figure 167 on [page 318](#)).

Note: Do not use variables for routing rules since routing rules can be used by more than one strategy and variables cannot.

Warning! Universal Routing 7.2 through 7.6 does not support Routing objects in strategies that use routing rules with targets of type Queue Group.



Default

Unlike other Routing objects, the Default object does not use a properties dialog box. Instead, the Default routing object tells URS to route the interaction to the default destination, as specified by the default DNs at the routing point or by the configuration option `default_destination` (see [page 638](#)).

When you use the Default object in a strategy, it sends the interaction to that destination. The Default object does not have an associated properties dialog box and you cannot assign properties to the Default object. Also see “Default Routed Calls” on [page 730](#).

Note: Default DNs are set through Configuration Manager as a property of a routing point.



Force Routing

This object uses a routing rule to tell URS to force the interaction to the first target type (ACDQueue, Destination Label, or Routing Point) without any other operations. [Figure 165](#) shows properties for a Force Routing rule that is routed to an ACD queue.

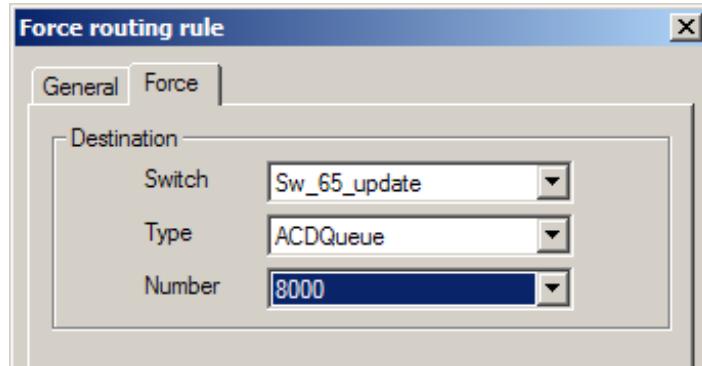


Figure 165: Force Routing Rule Dialog Box, Force Tab

The interaction is routed unconditionally; URS does not check the status of the destination. This object is similar to the Force function in the Function object. See [page 362](#) for additional information on this object.

Warning! Force should always be thought of as a last plan of action and therefore used infrequently.



Load Balancing

Note: This object is used for load balancing between queues. For information on load balancing between multiple URS, between targets, and IVR server load balancing, see the chapter on load balancing in the *Universal Routing 8.1 Deployment Guide*. Also see Router Self-Awareness in that guide where URSs deployed in a load sharing mode can communicate with each other.

This object uses a preconfigured routing rule (see “Creating Routing Rules for Routing Objects” on [page 312](#)) to distribute voice interactions to URS queues according to estimated wait time (see [page 721](#)). [Figure 166](#) shows the properties dialog box for the Load Balancing Routing object in.

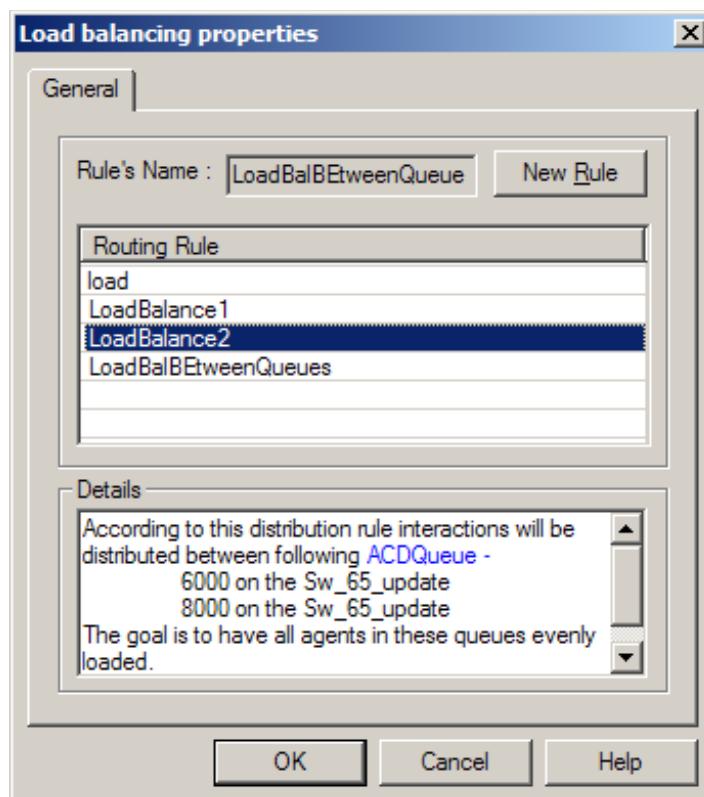


Figure 166: Load-Balancing Properties Dialog Box

See Chapter 7, “Routing Statistics,” [page 715](#) and “User-Defined Statistics” on [page 732](#), as well as the *Reporting Technical Reference Guide for the Genesys 7.2 Release*, for more detailed descriptions of available contact center statistics. For information about load balancing, see the respective chapter in the *Universal Routing 8.1 Deployment Guide*.

Warning! A Share Agent by Service Level Agreement (SLA) solution or a Cost-Based Routing solution, as described in the *Universal Routing 8.0 Routing Application Configuration Guide*, are not compatible with Load Balancing routing rules. The additional target selection criteria specified in an SLA solution, or using cost as additional target selection criteria, will upset the balance. For additional information, see the warning on [page 722](#).



Percentage

Use the Percentage object to route voice interactions. This object uses a preconfigured routing rule to distribute interactions to targets based on a set percentage for each target (see “Creating Routing Rules for Routing Objects” on [page 312](#)). [Figure 167](#) shows the properties dialog box that opens when you double-click the Percentage Routing object in a strategy.

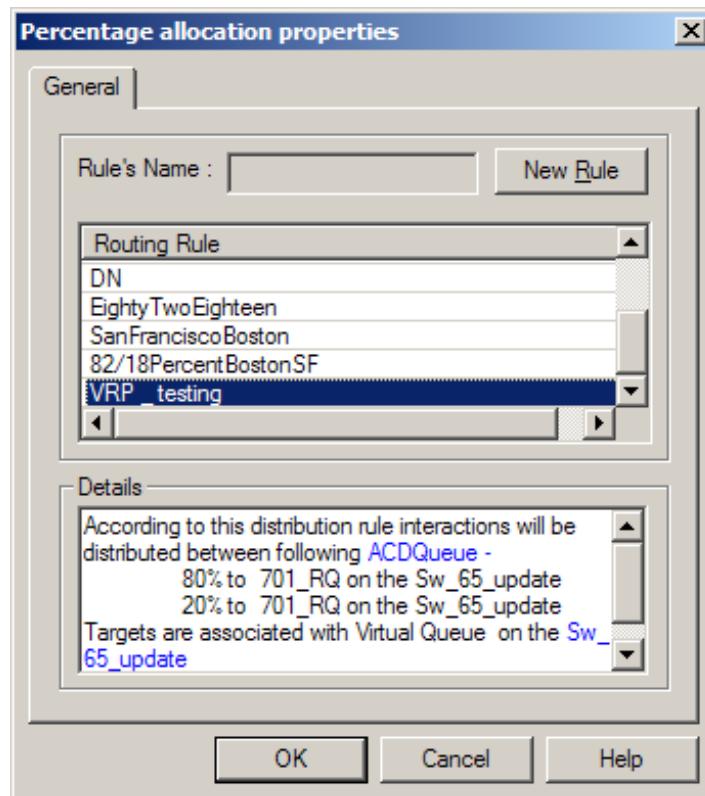


Figure 167: Percentage Allocation Properties Dialog Box

When this criterion is used, every target (see Figure 160 on [page 313](#)) must be supplied with a non-negative integer percentage. Targets with a percentage of 0 can only receive interactions when no target with a non-zero percentage is available.

When URS distributes interactions to targets on such target lists, it attempts to distribute to each target a proportion of all interactions arriving to the target list. It attempts a distribution that equals the ratio of the target's own set percentage to the sum of all percentages of targets on the list (the ideal ratio). URS achieves this by routing the interaction to the available target for which the proportion of received interactions is running the farthest behind the ideal ratio.

To complete the Percentage tab of a Percentage routing rule (see Figure 160 on [page 313](#)):

1. From the Type drop-down menu, select a type (ACDQueue, Agent, Agent Group, Destination Label, Place, Place Group, Queue Group, or Routing Point).
2. Using the Add Item button, add the number of targets desired.
3. For each target, in the Destination Name column, click the arrow to display a dialog box and select a value for the target.
For Agent, the value is the employee ID. For Agent Group, Place, and Place Group, the value is the name of the corresponding object in Configuration Layer. For Routing Point, ACDQueue, and Destination Label, the value is the Alias of the corresponding DN in Configuration Layer. (The target type in this dialog box is preselected and no other type can be selected in this dialog box.)
4. In the% column, specify a percentage for each target.

Important Notes

- Interaction Routing Designer does not limit the percentage to 100. If the sum of the percentages for the listed targets is greater than 100, all percentages adjust so that the total equals 100 percent.
- URS supports using non-configurable targets in percentage routing rules in the format <DN_number>@<switch_name>. This differs from the non-configurable targets used in functions, which use the format <DN_number>@<switch_name>.DN.
- The Percentage routing rule is most appropriate to use with targets that URS always assumes to be ready, such as ACD queues, route points, and so on. If you use Agent, AgentGroup, Place, or PlaceGroup targets, the specified percentage distribution is not guaranteed in a situation where there is no available target for the interaction.
- Cost-Based Routing (CBR), as described in the *Universal Routing 7.6 Cost-Based Routing Configuration Guide*, is incompatible with Percentage distribution. If CBR and Percentage distribution are both activated, then URS uses the Percentage object and CBR is disabled for the call.

- A Share Agent by Service Level Agreement (SLA) Routing solution, as described in the *Universal Routing 8.0 Routing Application Configuration Guide*, is also incompatible with Percentage distribution. The additional target selection criteria specified in an SLA solution will upset the balance. For additional information, see the warning on [page 722](#).



Queue Interaction

Notes: See “About Genesys Queues” on [page 150](#). *Universal Routing 7.6 Strategy Samples* contains many examples of how to use this object.

When a non-voice strategy is part of a business process (see [page 149](#)), you must always finish the strategy and each strategy branch with one of three routing objects: Stop Interaction (see [page 274](#)); Route Interaction (see [page 322](#)); or Queue Interaction (this page).

Queue Interaction Strategy-Linked Nodes

The specified interaction queue flows out of the right side of a Strategy object in a business process. [Figure 168](#) shows a routing strategy (Inbound e-mail preprocessing st) that uses four queues (Inbound e-mail postprocessing, Collaboration reply e-mails, E-mails route to original agent and Inbound e-mail failure).

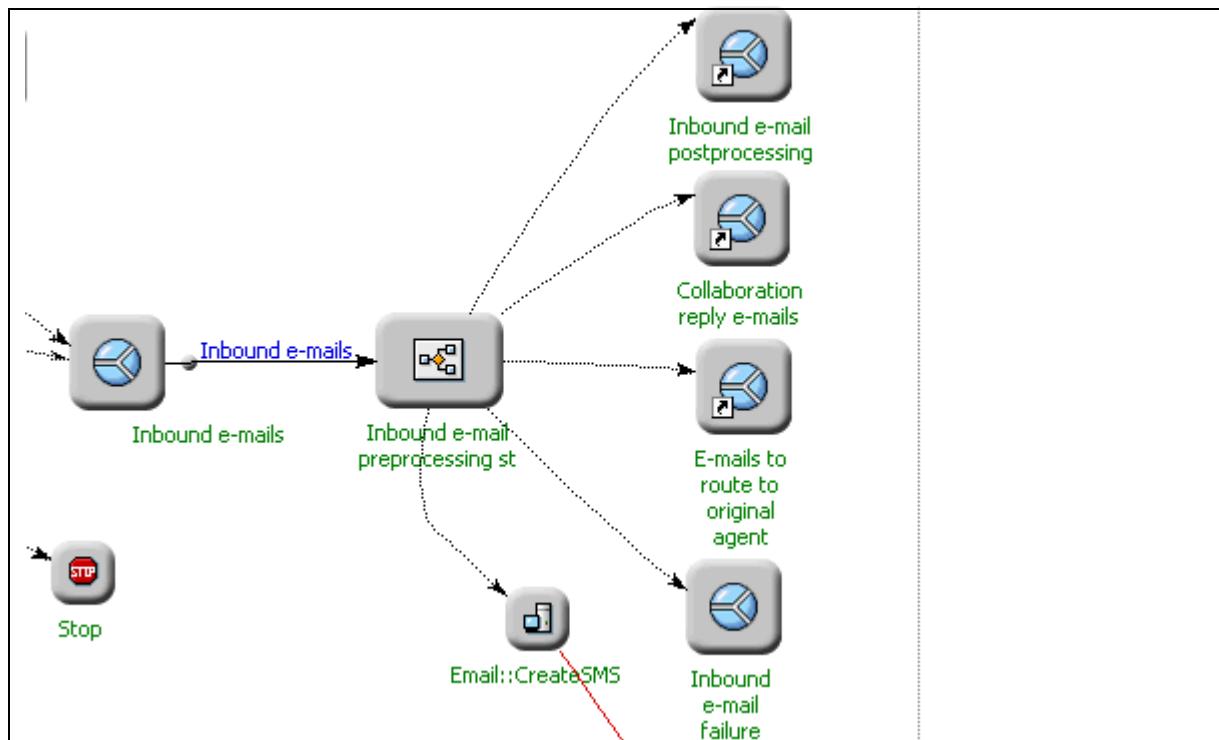


Figure 168: Queue Objects Generated by Strategy Object



Queue Interaction General Tab

Figure 169 shows the Queue Interaction Properties dialog box after clicking the Interaction queue down arrow to show available queues configured in the Interaction Design window.

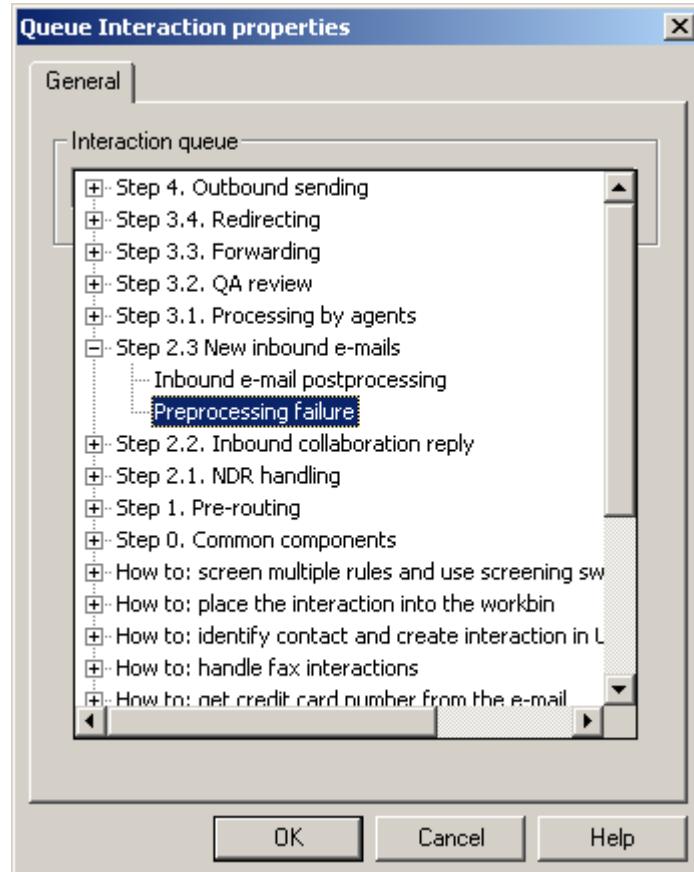


Figure 169: Queue Interaction Properties Dialog Box

As can be seen above, queues are organized by business process.

Note: Along with the business processes, the list box in Figure 169 contains a predefined `Independent Objects` folder that contains an interaction queue called `Same queue(_BACK_)`. This is the only predefined queue for the Queue Interaction object.

The Queue Interaction Properties dialog box shown in Figure 169 has only one parameter: the name of the queue. Click the down arrow and select the interaction queue (see “About Genesys Queues” on page 150).



Route Interaction

Note: The Route Interaction object can be used for both chat (Interaction Server) and instant messaging (SIP Server) interactions.

The Route Interaction object enables you to select one or more target objects: Agent, Agent Group, Campaign Group, Place, Place Group, Skill, or Variable.

Optional parameters include input and output queues configured in the Interaction Design window (see Figure 46 on [page 100](#)).

[Figure 170](#) shows both the Interaction Queue and Target Selection tabs as well as the dropdown menus for Queues and Type.

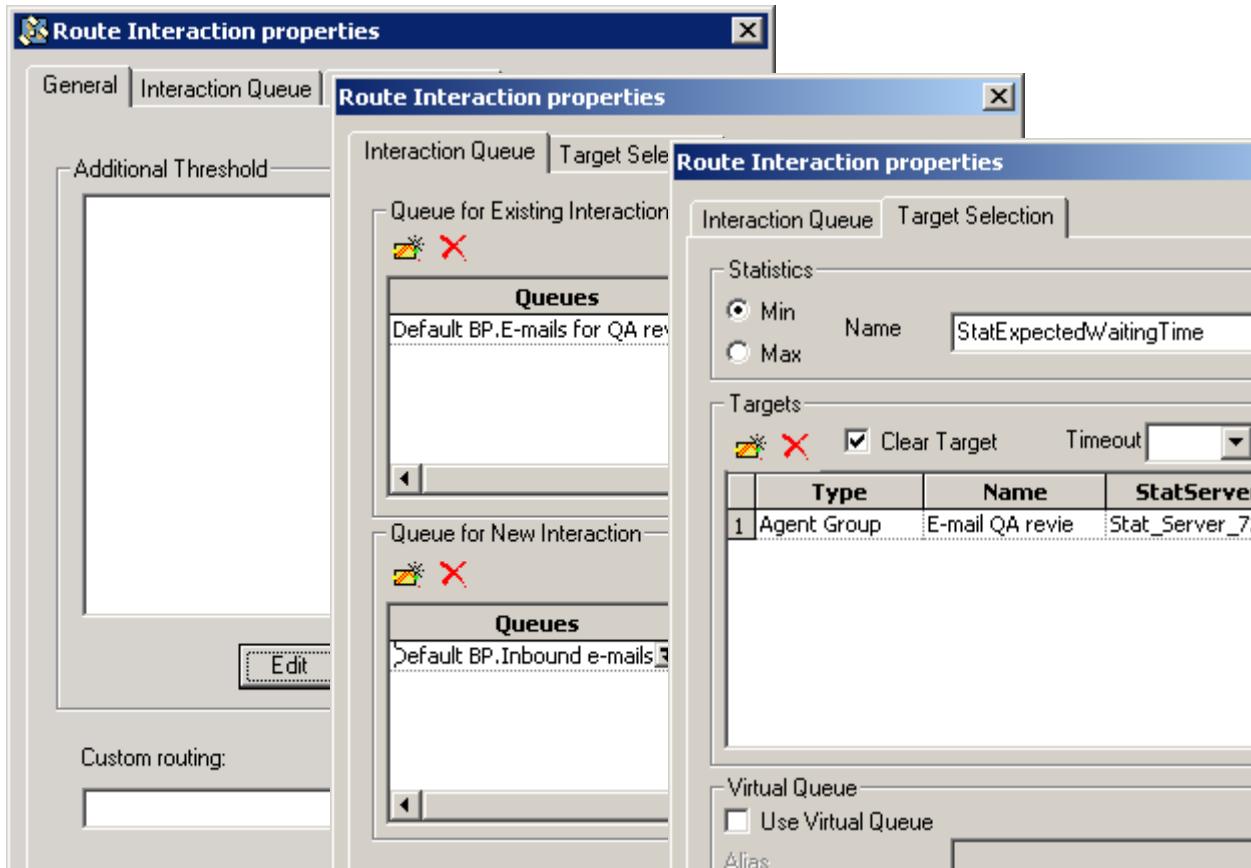


Figure 170: Route Interaction Properties Dialog Boxes

The Target Selection tab is required. The Interaction Queue tab is optional.

As shown in Figure 168 on [page 320](#), when the Interaction Queue tab is used, one or more queue nodes are linked to the strategy in a business process.

Stop and Target Strategy-Linked Nodes

The Route Interaction object causes one or more strategy-linked nodes to appear in a business process (see “Business Processes” on [page 149](#)). If you select the Stop_Processing (_STOP_) queue, a Stop node appears. For example:

- An Agent is selected as routing target in the Target Selection tab.
- Stop_Processing is selected under Queue for Existing Interaction.

As shown in [Figure 171](#), the agent target (Dynamic target) and Stop nodes are linked to the strategy in the business process.

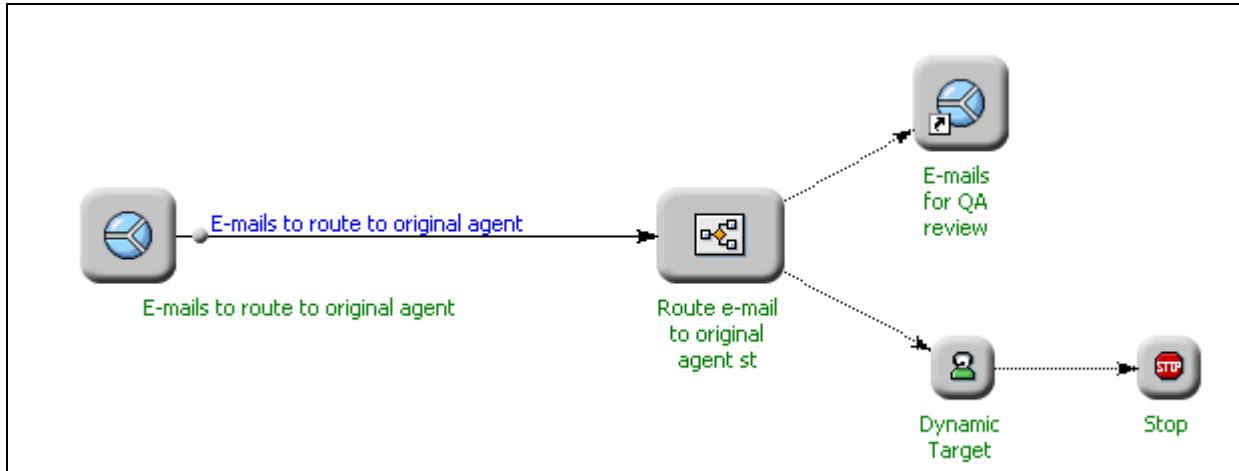


Figure 171: Strategy-Linked Nodes In a Business Process

Route Interaction Properties Dialog Box

This section explains how to complete the `Interaction Queue` and `Target Selection` tabs in the `Route Interaction Properties` dialog box shown in [Figure 170](#) on [page 322](#).

General Tab

Beginning with release 8.1.2, the new Custom Routing functionality allows automatic splitting of code functionality. This splits target selection of objects into a set of functions (`SelectDN`, `SuspendForDN`, `RouteCall`) under the cover of single target selection object. This allows any code to be put in a strategy between a target selection and routing. Additional code can be inserted before the `RouteCall` function is provided in the form of subroutine.

Fill in the `Custom routing` parameter shown in [Figure 170](#) to activate the custom routing functionality. Based on that, Routing Selection and Route Interaction objects provide the possibility of performing “Custom Routing”. Custom routing is a subroutine that a user can provide in the properties of Selection and/or Route Interaction object. If provided, the subroutine is invoked after the target selection, but before routing. It allows users to

intercept, report, and change the information about the selected target. As well, the routing can be canceled.

Subroutines must have one input and one output parameter. The Input parameter describes

the selected target in the same format as the functions `SelectDN/SuspendDN` uses. As well, subroutines can analyze a selected target, and implement any attaching of reporting/data.

The subsequent Interaction processing is determined by the returned value of the subroutine:

- Return the provided target unchanged - In this case, the call is just routed to the selected target.
- Replace some parameters of the selected target (for example: dn) - In this case, the call is routed to the updated destination.
- Raise an error (if the wrong target is selected) - In this case, the interaction is not routed, and control flow goes through the red port of a target selection object.
- Return an empty string - In this case, URS assumes that the subroutine itself performs the routing, and there is no need to do anything else with interactions. The flow just goes through the green port of the target selection object.

If Selection (Route Interaction) objects have no custom routing subroutines specified (default), the objects run normally.

Interaction Queue Tab

Use the information in [Table 105](#) to complete the optional Interaction Queue tab:

Table 105: Route Interaction Object, Interaction Queue Tab

Parameter	Description
Queue for Existing Interaction	Click the button to add a new entry ().
	Click in the empty Queues text box and a down arrow appears. Click the down arrow and select an interaction input queue previously defined in Interaction Design window (see page 98). The Stop Processing value is predefined.
Description	Click in the empty text box and a down arrow appears. Click the down arrow and enter or select a description, which will be used as a hint on the agent desktop. For example, after the agent constructs a response, the response may need to be checked by QA. In this case, the description would prompt the agent to place the interaction in the queue read by QA.

Table 105: Route Interaction Object, Interaction Queue Tab (Continued)

Parameter	Description
	If there is more than one queue, repeat Steps 1~3.
New Interaction	Repeat the above process to define interaction output queues to be used when a new interaction is created.

Target Selection Tab

Use the information in [Table 106](#) to complete the Target Selection tab:

Table 106: Route Interaction Object, Target Selection Tab

Parameter	Description
Statistics	Optional. If more than one target is available from the target list when an interaction encounters the Route Interaction object, URS uses this Statistics information. Also see “List of Predefined Statistics” on page 719 . <ul style="list-style-type: none"> Note: When routing to a target that is a meta-object (such as a Group of Agents or a Group of Places) and using any Routing object statistic except StatAgentLoading and StatAgentLoadingMedia, URS does not expand the target to its underlying level (for example, the underlying Agents belonging to the Group). Instead, URS applies the statistic directly to the meta-object.
Min Max	Click the Min or Max button to indicate whether URS should use the minimum or maximum value of the statistic to be selected next. See page 332 for information about the StatAgentLoadingMedia statistic.
Name	Click the down arrow and select the statistic to be used when more than one target is available.
Targets	Click the button to add a new entry () to enable fields. Your strategy will not compile unless a target is specified.
Type	Click in the empty Type text box and a down arrow appears. Click the down arrow and select the target type (Agent, Agent Group, Campaign Group Place, Place Group, Variable, or Skill). For more information, see “Statistical Objects (Target Types)” on page 358 .

Table 106: Route Interaction Object, Target Selection Tab (Continued)

Parameter	Description
Name	<p>Click in the empty text box and a down arrow appears. Click the down arrow and select the specific Agent, Agent Group, Campaign Group, Place, Place Group, Variable, or Skill. If you selected Skill as the target type, the Skill Expression Properties dialog box opens where you can construct a skill expression (see Figure 40 on page 89). See “Logical Expressions” on page 87 for help in constructing a skill expression.</p> <p>Note: Stat Server currently supports virtual group functionality with two types of agent parameters: Skill configured for an agent and ACDQueue (voice routing) to which an agent is logged in. You can simultaneously specify both types of parameters in an expression for a single virtual group. For more information, see the Virtual Agent Groups chapter in the <i>Framework 8.1 Stat Server User’s Guide</i>.</p>
Stat Server	Click in the empty Stat Server field and a down arrow appears. Click the down arrow and the Key dialog box appears. In the Key dialog box, click the Value down arrow and select the name of the Stat Server to supply statistical information used for routing.
	Optional. If URS should consider more than one target after the Timeout number of seconds, repeat the above steps.
Clear Target	Optional. Click the check box if targets listed in the object should be retained after the interaction moves on through the strategy and encounters other Selection object.
Timeout Sec	Optional. Enter the timeout number of seconds to specify the time an interaction waits for an available target. If the timeout expires, the interaction is routed to the red port.
Use Virtual Queue	Set a Virtual Queue for by specifying the Alias, Switch, and Number. See important note on page 333 .

Note: IRD provides a warning if the timeout field exceeds the maximum value that can be processed (the maximum value, if multiplied by 1000, should not result in an overflow to 32 bit integer values).



Selection

Use this object to route voice interactions as well as to route instant messaging (SIP Server) interactions as described in the *Genesys 8.1 Instant Messaging Solution Guide*. To route chat (Interaction Server) interactions, use the Route Interaction object ([page 322](#)).

General Tab

The Selection object has a General tab, which can be used for expressions. One type of expression that can be used here is a threshold expression that contains the conditions for borrowing and lending agents in a Share Agent by Service Level Agreement routing solution. [Figure 172](#) shows an example expression created by clicking the Edit button to open the Threshold Expression Properties dialog box containing a threshold expression.



Figure 172: General Tab in Routing Selection Object

For information on creating threshold expressions, see the *Universal Routing 8.0 Routing Application Configuration Guide*.

Note: Threshold expressions support all arithmetic operations: +, -, *, /. Threshold expressions are used as parameters for the functions SetTargetThreshold or as the Threshold value in the Target Selection object.

Custom Routing

Beginning with release 8.1.2, the new Custom Routing functionality allows automatic splitting of code functionality. This splits target selection of objects into a set of functions (SelectDN, SuspendForDN, RouteCall) under the cover of single target selection object. This allows any code to be put in a strategy between a target selection and routing. Additional code can be inserted before the RouteCall function is provided in the form of subroutine.

Fill in the `Custom routing` parameter shown in [Figure 173](#) to activate the custom routing functionality.

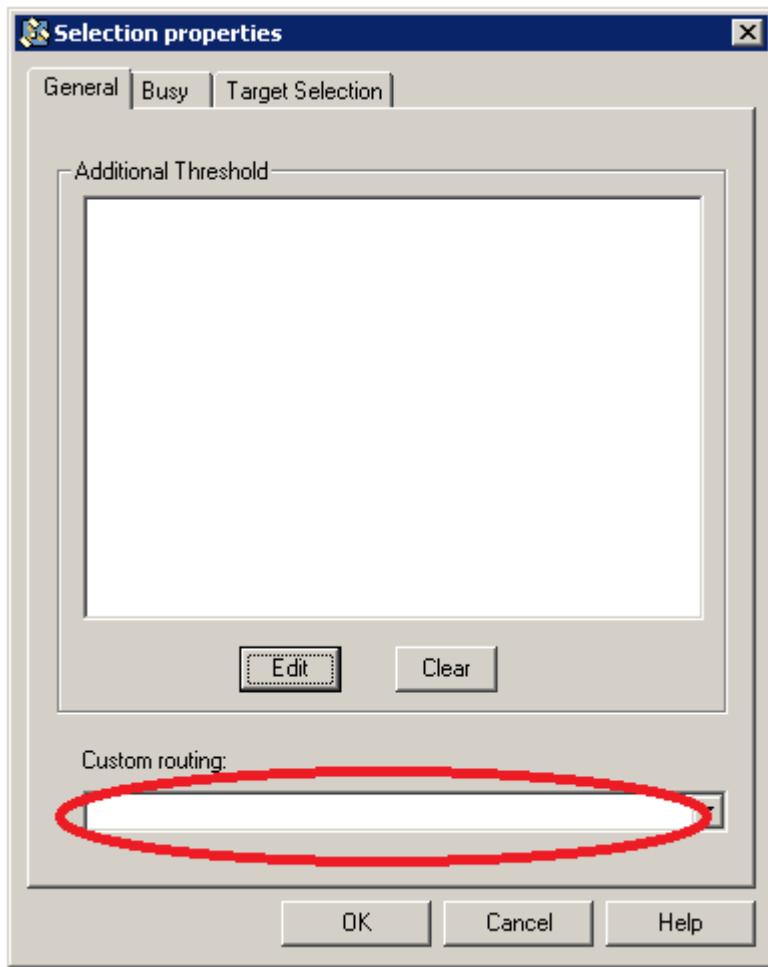


Figure 173: Custom Routing

Based on that, Routing Selection and Route Interaction objects provide the possibility of performing “Custom Routing”. Custom routing is a subroutine that a user can provide in the properties of Selection and/or Route Interaction object. If provided, the subroutine is invoked after the target selection, but before routing. It allows users to intercept, report, and change the information about the selected target. As well, the routing can be canceled.

Subroutines must have one input and one output parameter. The Input parameter describes

the selected target in the same format as the functions `SelectDN/SuspendDN` uses. As well, subroutines can analyze a selected target, and implement any attaching of reporting/data.

The subsequent Interaction processing is determined by the returned value of the subroutine:

- Return the provided target unchanged - In this case, the call is just routed to the selected target.

- Replace some parameters of the selected target (for example: dn) - In this case, the call is routed to the updated destination.
- Raise an error (if the wrong target is selected) - In this case, the interaction is not routed, and control flow goes through the red port of a target selection object.
- Return an empty string - In this case, URS assumes that the subroutine itself performs the routing, and there is no need to do anything else with interactions. The flow just goes through the green port of the target selection object.

If Selection (Route Interaction) objects have no custom routing subroutines specified (default), the objects run normally.

Note: IRD provides a warning if the timeout field exceeds the maximum value that can be processed (the maximum value, if multiplied by 1000, should not result in an overflow to 32 bit integer values).

The parameters of the Selection object also include target selection criteria as well as a set of Busy Treatments (see [page 366](#)) that provide instructions for handling the interaction before an eligible target is available to receive it. [Figure 174](#) shows available target types in the Target Selection tab and available treatment types in the Busy tab.

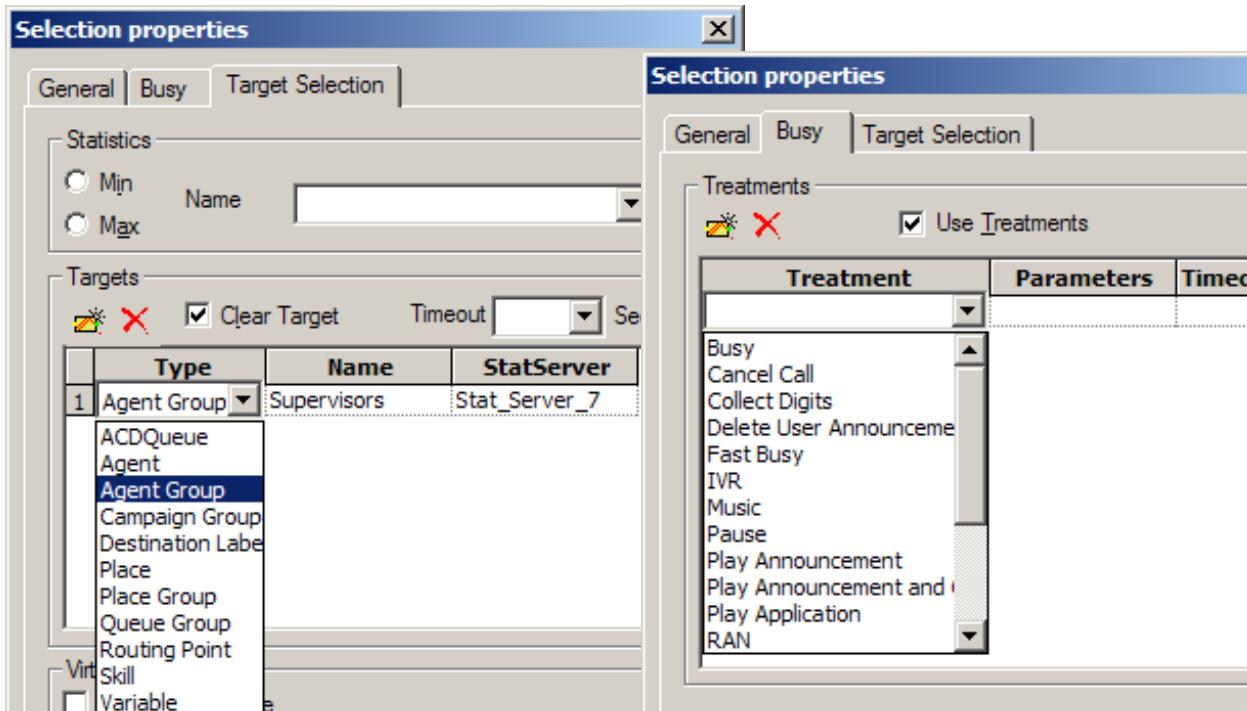


Figure 174: Busy Tab and Target Selection Tab in Selection Object

For information on the Busy tab, see “[Busy Treatments in Routing Objects](#)” on [page 366](#).

Target Selection Tab

Specify one or more targets as needed for this object using the button to add an item on the Target Selection tab. For each target, do the following:

1. Select the target Type (ACDQueue, Agent, Agent Group, Campaign Group, Destination Label, Place, Place Group, Queue Group, Routing Point, Skill (see “Skill Target in Routing Objects” on [page 363](#)), or Variable). Also see “Statistical Objects (Target Types)” on [page 358](#).
2. In the Name column, click the down arrow to display a dialog box for that type.
3. For all types but Skill, select a value from the drop-down menu, and click OK. For Skill, click the down arrow to open the Skill Expression Properties dialog box, which allows you to create a skill expression (see Figure 40 on [page 89](#)).

Note: Stat Server currently supports virtual group functionality with two types of agent parameters: a Skill configured for an agent and an ACD Queue to which an agent is logged in. You can simultaneously specify both types of parameters in an expression for a single virtual group. For more information, see the Virtual Agent Groups chapter in the *Framework 8.1 Stat Server User’s Guide*.

Warning! Skill expressions cannot exceed 100 elements (skill names, numbers, comparisons, and logical operands); that is, a skill expression should have no more than 25 constructions, such as English > 1. Also, the maximum size of an overall skill expression (as text) cannot exceed 10239 bytes.

4. Select a Stat Server from the drop-down list in the Stat Server column for all types but Variable. For the type Variable, a Stat Server must be defined for each target assigned to the variable.

See “Targets in Other Objects” on [page 373](#) for more information on target types. Also “Using a Skill Expression to Exclude an Agent” on [page 366](#).

If more than one target is available from the target list when an interaction encounters the Selection object, URS uses the Statistics and Stat Server information (see Figure 174 on [page 329](#)) to determine the target.

This object also includes the following options:

- Clear Target—sets whether the targets listed in the object are retained after the interaction moves on through the strategy and encounters other Selection objects. This check box automatically generates the same code as the ClearTargets function described on [page 510](#).

If **Clear Target** is not selected and the current interaction encounters Selection object(s) later in the strategy, the targets in this Selection object are added to those of the next Selection object(s). If **Clear Target** is selected, the targets are not added to the list of any Selection objects that the interaction encounters later in the strategy.

For example, if two Selection objects are used in a strategy with the first Selection object configured with three agents and a timeout of 30 seconds and the second Selection object configured with four new agents and a timeout of 60 seconds, the **Clear Target** setting determines which agents are considered as targets for an interaction.

When the **Clear Target** option is selected in the first Selection object, URS waits for 30 seconds for any of the three agents to become available. If none are available before the timeout expires, the strategy resumes. When the second Selection object is encountered, URS only considers these four new agents as possible targets during the 60-second timeout. URS does not consider (clears) the first three agents as possible targets.

When **Clear Target** is not selected, and none of the agents listed in the first Selection object are available, when the interaction encounters the second Selection object, URS considers the first three agents and the four new agents as possible targets during the 60-second timeout set in this object. If no more Selection objects follow the second Selection object and no agents are available before the timeout expires, the interaction is routed to the red port.

Note: This option is only applicable if more than one Selection object is used along the same path in a strategy. It has nothing to do with how agents within the object are evaluated for availability. If no further Selection objects are included in the strategy and no agent is available, the interaction is default-routed. If more than one agent is available, the statistic specified in the **Name** field (see Figure 174 on page 329) is used to determine which agent receives the interaction.

- **Timeout**—specifies the time an interaction waits for an available target. The **Timeout** field accepts a value of up to 2147483 seconds. If the value is not specified, it defaults to 0 (zero). When the timeout expires before one of the targets is available, the interaction is routed to the red port. This value can be specified as a variable (previously defined in the **Variable List** dialog box) or an integer.

Note: Upon entering any target selection object, if a target was not selected and at least one statistic is not open and waiting time is 0, URS adjusts waiting time to the maximum of `reservation_pulling_time` and `treatment_delay_time` values. This eliminates the possibility of not routing the first call after URS is started.

- **Statistics Min, Max, Stat Server Name**—used by Universal Routing Server to determine which target to route the interaction to if more than one target is available. Select Max or Min to specify whether the interaction should be routed to the target with the maximum or the minimum value of the statistic. Also see “List of Predefined Statistics” on [page 719](#).

Agent capacity rules, set in the Genesys Agent Capacity Wizard, provide information about whether an agent is available for routing. After defining a complete set of available agents (taking agent capacity rules into consideration, if configured), URS applies the selection criteria specified in the Routing objects. This can include using the minimum or maximum value of a statistic when using the Statistics, Route Interaction, or Selection Routing objects.

However, if the `StatAgentLoading` or `StatAgentLoadingMedia` statistic (see [page 724](#)) is selected in one of these Routing objects, then URS calculates it in a special way using an agent state vector supplied by Stat Server as a part of the Agent Status structure.

Warning! If configuring Skills or Agent level routing and your site has a large number of agents, Genesys recommends using the `StatAgentLoading` (see [page 722](#)) or `StatAgentLoadingMedia` statistic (see [page 724](#)) to select agents. Using the deprecated `StatTimeInReadyState` statistic can result in high CPU usage even when there are no calls.

Note: When routing to a target that is a meta-object (such as a Group of Agents or a Group of Places) and using any Routing object statistic except `StatAgentLoading` and `StatAgentLoadingMedia`, URS does not expand the target to its underlying level (for example, the underlying Agents belonging to the Group). Instead, URS applies the statistic directly to the meta-object.

- **Virtual queue** (if desired)—sets the virtual queue for the object. The virtual queue is not a physical queue but rather a logical queue to which interactions are queued if the specified targets are not available. From the drop-down lists select the Alias, Switch, and Number for the queue. For Alias, you can also select predefined variables from the list.

Notes: When you have two Switch objects with the same name created respectively in Environment and in some Tenant, you will not be able to select virtual queues created for the Environment's Switch while working with Selection, Route interaction, and Workbin strategy objects in the Tenant. Both Switches with the same name will display in the corresponding drop down box, but each of them will display virtual queues created only for the Tenant's Switch.

URS supports using non-configurable targets in the Selection object (using variables) in the format <DN number>@<switch_name>.DN.

Functions: SelectDN (see [page 608](#)) and SuspendForDN (see [page 614](#)) are alternatives to the Routing Selection object. They provide the equivalent functionality to this object.



Service Level

Warning! A Share Agent by Service Level Agreement (SLA) solution or Cost-Based Routing solution, as described in the *Universal Routing 8.0 Routing Application Configuration Guide*, is not compatible with the use of Service level routing rules. The additional target selection criteria will upset the balance.

Use this object to route voice interactions. The Service Level object enables you to select or create a routing rule to specify a Service Factor for a customer segment. For example, you may want to distribute 60% of interactions in less than 10 seconds to a specific agent group. In addition, you can further refine target selection by specifying Best Fit skills and by selecting the minimum or maximum value of a statistic for a Stat Server. Agents with this minimum or maximum value are selected.

Note: Skill expressions (used by the Selection and Service Level Routing objects) cannot exceed 100 elements (skill names, numbers, comparisons, and logical operands), that is, a skill expression should have no more than 25 constructions, such as English > 1. This limitation is especially important to observe with Service Level routing rules, which use skills internally. Every skill criteria used in a Service Level routing rule generates 1-3 constructions like Skill > Number. Also, the maximum size of an overall skill expression (as text) cannot exceed 10239 bytes.

After creating a Service Level routing rule (see Figure 159 on [page 312](#)), you can select it in the properties dialog box as shown in [Figure 175](#).

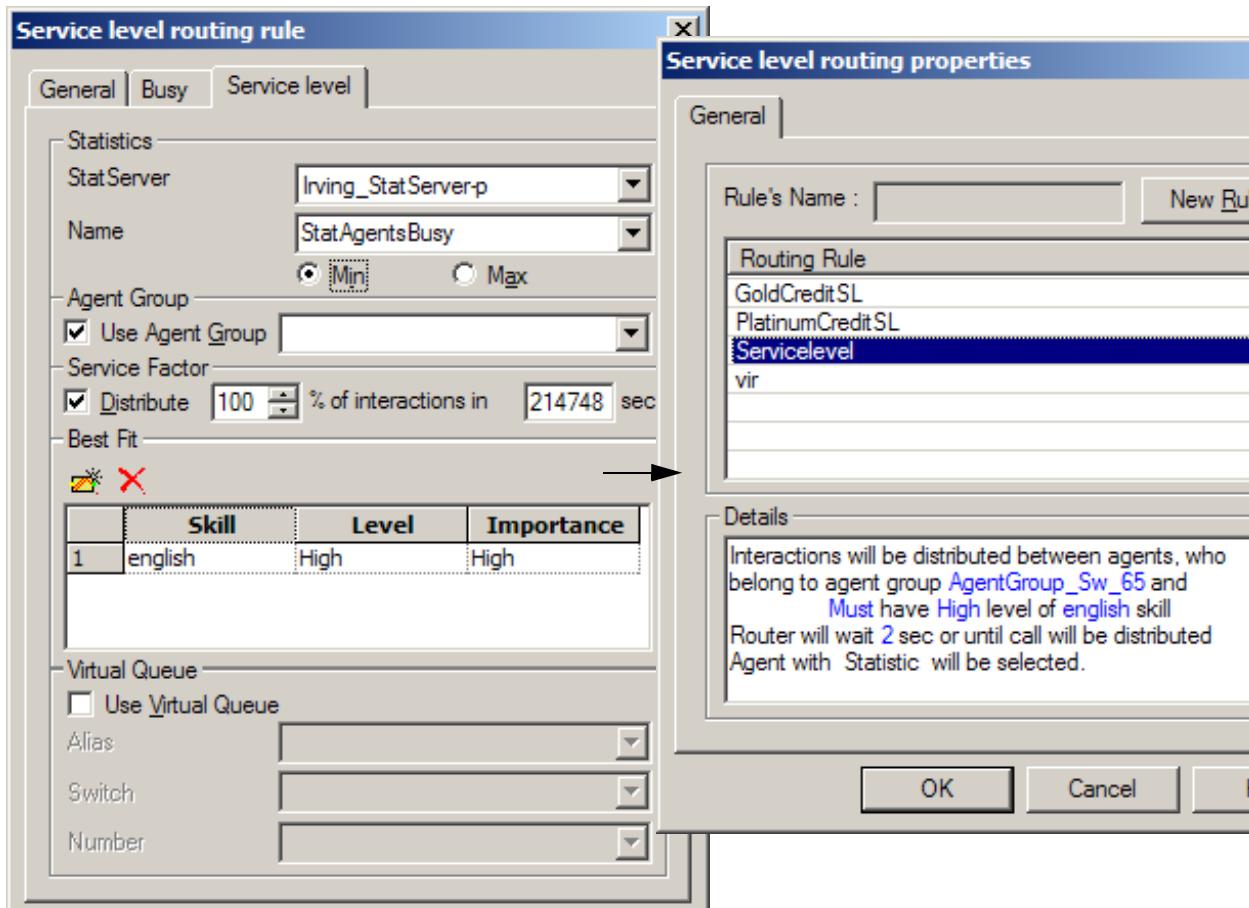


Figure 175: Service-Level Routing Rule

After you click OK in this dialog box, the Service level routing properties dialog box in [Figure 175](#) details the routing rule.

When Service Factor is not selected, this object functions like a skills-based Routing object.

Important Information on Service Level Routing

- In the Service Level routing rule algorithm, the expansion of the current working set of agents from ideally-skilled to non-ideally skilled is no longer purely based on 30-second calls or 50 ended calls (which ever comes first). The algorithm now considers calls in queue, especially applicable in a low volume and/or long talk time contact center.
On top of the current regulating mechanism (call it the *primary* mechanism) activated by calls quitting (successfully or not), a *secondary* regulating mechanism works in following way:
When call wait time has exceeded the Service Factor threshold, a 30 second timer is activated. If during the next 30 seconds, the primary mechanism does not cancel it, the secondary mechanism goes into effect. It

increases the current working set of interactions in the same way as if ServiceLevel factor was not in effect for this routing rule.

Note: The secondary mechanism works only when primary mechanism is not activated (activating the primary mechanism cancels the secondary one).

- If there are sudden increases or decreases in the number of interactions, URS may require more than one interval to adjust the set of agents to maintain the Service Level.
- Because URS only calculates the Service Factor at intervals of 50 interactions or 30 seconds, Service Level should not be used for real-time reporting but only for debugging purposes.
- Use the Service Level object with care. The adjustment that URS makes to expand or contract the set of agents to maintain the Service Level could result in high-priority interactions being received by a less skilled agent.
- Because priorities are assigned to interactions for strategies using Service Level rules, you cannot assign external priority to the interaction.
- URS can report on the service level being achieved using the automatic peg PegSF. The achieved service level can be viewed in CCPulse+ and CC Analyzer. However, as mentioned previously in “Important Information on Service Level Routing” on [page 334](#) for Service Factor, Service Level should not be used for real-time reporting but only for debugging purposes.
- Do not load two different strategies using the same set of agents to routing points handled by the same URS when only one of the strategies contains a Service Level object. The priority of an interaction waiting in queue increases at regular intervals for a strategy with a Service Level object but does not increase for interactions waiting in queue for a strategy without a Service Level object. This will affect the order in which interactions are routed.
- For a comparison of Service-Level routing versus routing based on Service Objectives, see [page 681](#).
- For strategies containing a Service-Level routing rule, when URS calculates the service level to determine whether a specified level is met for interactions entering virtual queues, the calculation is based on the interval that starts with EventQueued and ends with EventDiverted. This value is translated into the CallDistributed statistic by Stat Server. When generating reports using CCPulse+ or CC Analyzer, the statistic CallAnswered is used instead, which is based on a longer interval starting with the time that the interaction enters the routing point and ending with EventEstablished.

Service Level Timers

In URS memory, each Service Level Routing Object (SL object) has an array of 50 entries, which is initially empty. When call processing is complete for an SL object (meaning strategy control for the call has gone out the green or red port of the object), URS records, in the next free element of the array, if Service Level processing was successful or not.

- If the call was routed within the time specified in the rule, then processing was successful.
- If routing required more time or the call was not routed at all (it went through the red port), then processing was not successful.

Primary 30-Second Timer for Procedure 1

If the recorded element is the first element in the array, then URS activates a 30-second timer for this SL object. After this, SL object checking will happen either on expiration of the 30 second timer or when the array becomes full (whichever is first). The checking procedure performs tasks such as:

- Counting the percentage of successfully processed calls,
- Adjusting the working list of agents according to this percentage,
- Cleaning up (emptying) the array
- Resetting the 30-second timer.

The next invocation of the checking procedure does not happen at the 30-second timer unless there has been at least one call for processing by the SL object since the previous invocation of the procedure. The next invocation would not occur, for example, if a call arrived just after a 30-second interval during which there was no call. There could also be a situation of no Ready agents in the working list, with no opportunity to expand it for a long time. So this call and all subsequent calls could wait a long time without URS trying to expand the working list.

Secondary 30-Second Timer for Procedure 2

To overcome the above issue, a secondary 30-second timer is activated for the SL object. It triggers a procedure that does what the regular checking does if SL criteria are not met; it expands the working list of targets. However, after activating this timer, and before it expires 30 seconds later, a call could leave this SL object and, therefore, the regular (primary) 30-second timer has to be initiated. If this occurs, the secondary timer is cancelled.

Note: Remember, when a call leaves a rule, URS puts info about it in the next free element of the array. If this happens to be the first element in the array, then URS activates the primary timer. Doing this cancels the secondary timer activated previously. All subsequent calls leaving the rule will do nothing in this respect – only the first entry in the array activates the primary timer.

Summary

- There are two different procedures triggered by two different timers.
 - Procedure 1 (primary). If invoked, it evaluates whether the routing rule criteria are being met and increases, decreases, or leaves unchanged the current working list. Each time URS places the first element in the 50-element array, it sets a 30-second timer and activates Procedure 1 when the timer expires (unless no calls have come in to the SL object).
 - Procedure 2 (secondary). if invoked, it expands the current working list because it is only invoked when SL criteria are not satisfied.
- Every time some call waits more than the requested time, URS sets a 30-second timer and will activate Procedure 2 when the timer expires.
- Only one timer can be active at a time.
- Activating the timer for Procedure 1 (primary mechanism) cancels a timer for Procedure 2 (secondary mechanism), if there is one active.
- A timer for Procedure 2 is not activated if a timer for Procedure 1 is already active.
- If there are sudden increases or decreases in the number of calls, URS may require more than one interval to adjust the set of agents to maintain the Service Level.
- URS can report on the service level being achieved using the automatic peg PegSF (see [page 713](#)). The achieved service level can be viewed in CCPulse+ and CCAalyzer. However, because URS only calculates the service factor at intervals of 50 calls or 30 seconds, PegSF should be used for debugging purposes only.
- When a call is sent to a Service Level routing rule for the first time, and there are no agents Ready in the working list of ideal agents, this is the formula that is used to determine how long the call will wait before the working list is expanded for the call: $100 / X*Y$ (Service Factor setting (X percent in Y seconds)).
- It could be, however, that the working list will be expanded before this time expires (but, in this case for all calls waiting in the Service Level rule) in the event either the primary or secondary mechanism (Procedure1 or Procedure2) gets called before the $100 / X*Y$ time expires.

Best Fit Requirements for Service Level Routing

Best Fit requirements (see Figure 175 on [page 334](#)) determine what skills an agent needs, the skill level the agent must have with respect to that skill, and how important the skill is for handling an interaction.

- **Skill** refers to the knowledge required in a particular area to handle customer needs.
- **Level** is the degree of competency that an agent must have with the skill. Skill levels include H (High, 8–10), M (Medium, 4–7), L (Low, 0–3). See “Using a Skill Expression to Exclude an Agent” on [page 366](#).
- **Importance** establishes how important the skill and skill level are with respect to an agent for handling an interaction. The Importance settings include:
 - C (Critical)—if specified the agent must have this skill to be considered as a target.
 - H (High).
 - M (Medium).
 - L (Low).

URS views these skill requirements as the optimal skill set required of the agents to receive an interaction and evaluates each agent accordingly.

Agent Evaluation and Service Level Routing

To determine who among the available agents could receive an interaction, URS rates each agent according to the required skill set and the level of proficiency the agent has for the skill. (URS rates the agent when a strategy containing a Service-Level object is initially loaded to a routing point.)

Agents that match the skill set specified in the Service-Level object become the members of the *ideal* set of agents. As long as the Service Factor ([page 342](#)) is maintained, only these agents receive interactions routed through the Service-Level object.

If the Service Factor is not maintained, URS determines how to adjust the agent group to maintain the required service level by evaluating the available agents who do not fall in the ideal agent group.

For each agent, URS calculates a skill deviation value, representing how much the skill set of an agent deviates from the required skill set.

Deviation is calculated using the following formula:

`abs(skill boundary - level) * weight of importance = deviation`

where:

- The `abs` function is the absolute value of a number. In this case it is the absolute (non-negative) value of the numerical difference between the `skill boundary` and the `level` for a particular skill.

- The skill boundary is the high- or low-end value within the range of the skill level specified for the skill in the Service-Level object. The high-end value is used if the agent's skill level is higher than the high-end value. The low-end value is used if the agent's skill is lower than the low-end value.

For example, by definition the value range of a medium skill level is from 4 through 7. If an agent has a skill level of 8 for that skill, 7 is used as the skill boundary. If an agent has a skill level of 2 for that skill, 4 is the skill boundary.

- The agent skill level is the agent's level for that skill.
- The weight of importance is the value representing the importance of a skill. (High importance = 16; Medium importance = 4; Low importance = 1; Critical is not calculated because an agent must have a skill of critical importance for the agent to be considered as a target.)

Example of a Deviation Calculation

If the skill level is medium for the skill and an agent has a skill level of 0 for that skill, the equation becomes:

$$ab(4-0)*4=16$$

This means that the agent has a deviation of 16 from the required skill set.

Criteria for Selection of Agents

URS selects agents according to the following criteria:

- When the skill requirements include a skill of critical importance, URS only selects agents with that skill and skill level even if those agents do not have any of the other required skills.
- When the skill requirements do not contain any skills of critical importance, URS selects agents who have at least one nonzero level for the required skills and who are within the tenant to which the routing point belongs or within the agent group when specified in the Service Level object.

URS also prioritizes all agents according to their deviation value. Agents with a 0 deviation are selected as a part of the ideal agent list (mentioned previously). All other selected agents are prioritized starting with those with the least amount of deviation from the ideal agent and ending with the greatest deviation.

The example provided in the next section shows how URS chooses and prioritizes agents according to their skills and deviation values.

Note: When evaluating agents as possible targets, URS also takes into account changes to such things as an agent's skill set, skill level, the addition of another agent with the skill set specified in the Service Level routing rule, and so on.

Service Level Routing Skill Example

In a particular business situation, the Service Level skills are defined as shown in [Table 107](#):

Table 107: Required Skills

Skill Name	Level	Importance
Gold Service	H	C
English	M	M
Bronze Service	L	L

The available agents and their associated skill levels are configured and URS calculates their deviation from the required skills as shown in [Table 108](#).

Table 108: Agent Skills

Agent	Skill	Level	Selected as Potential Target	Deviation from Ideal	Total Deviation
Agent1	Gold Service	9	Yes	0	16
	Writing	5		-	
	English	0		4*4=16	
	Bronze Service	0		0	
Agent2	Gold Service	9	Yes	0	0
	Writing	2		-	
	English	7		0	
	Bronze Service	0		0	
Agent3	Gold Service	7	No		
	Writing	1		-	
	English	2			
	Bronze Service	5			

Table 108: Agent Skills (Continued)

Agent	Skill	Level	Selected as Potential Target	Deviation from Ideal	Total Deviation
Agent4	Gold Service	8	Yes	0	6
	Writing	2		-	
	English	8		1*4=4	
	Bronze Service	5		2*1=2	
Agent5	Gold Service	0	No		
	Writing	4			
	English	0			
	Bronze Service	0			

In this scenario, URS rates each agent as follows:

- Agent2 is a member of the ideal agent group because Agent2 has 0 deviation.
- Agent1 and Agent4 are selected as potential targets but are not a part of the ideal agent group because they deviate from the required skill set. These agents are prioritized according to their deviation, which means Agent4 (deviation = 6) will be selected as a target before Agent1 (deviation = 16).
- Agent3 and Agent5 are not selected because they do not have the critical skill of Gold Service, which is why no deviation value was calculated.

If the Importance requirement was set to high for Gold Service rather than critical, Agent3 would have a Gold Service deviation of 16, an English deviation of 8, and a Bronze deviation of 2 for a total deviation of 26. This would make Agent3 the last agent selected as a possible target. Agent5 still would not have been considered as a target because the agent has 0 levels for all of the required skills.

Note: When a skill is defined as Low, agents with a skill level between 0 and 3 are selected. The skill level for agents without a level for a particular skill is understood as 0. When URS analyzes available agents based on the Service Level settings, no agent with all skill levels set to 0 is selected as a target. This guarantees that an agent must have some skills level above 0 to be included as a possible target.

Target Evaluation and Service Level Routing

Once URS rates all agents and prioritizes them, URS is ready to process interactions. As interactions arrive at routing points, URS routes them to agents in the ideal agent group who are available, while simultaneously determining that the service level is maintained.

If URS determines that the service level has dropped below the required level, URS examines the deviation value for all available agents to decide who should be added to the available agents. URS adds as many agents as necessary to return the Service Factor to the specified level.

Service Factor and Service Level Routing

Service Factor (see Figure 175 on [page 334](#)) is the percentage/interval pair that specifies that a certain percentage of interactions must be handled in a certain period of time, for example, 80 percent of interactions in 20 seconds.

At an interval of every 50 interactions or 30 seconds, URS calculates the Service Factor for that interval, based on the interactions that passed through the strategy. Using this information, URS decides whether to expand or contract the current set of agents available for the next interval.

At these intervals and based on the Service Factor setting (X percent in Y seconds), URS calculates three values:

- **SLReal**—the percentage of calls distributed in Y seconds
- **SLWarn**—the percentage of calls distributed in $3/4*Y$ seconds
- **AWT**—the average waiting time for distributed calls

URS uses **SLReal**, **SLWarn**, and **AWT** to determine whether to adjust the current group of target agents as follows:

- If **SLReal** equals the Service Factor percentage, URS makes no change to the current working agent group.
- If **SLReal** falls below the specified service factor ($SLReal < X$), URS compares the current **AWT** with the **AWT** previously measured.
 - If the current **AWT** is less than the previous **AWT**, the service factor is improving. URS assumes that the service factor will continue improving and makes no change to the current working agent group.
 - If the current **AWT** is greater than the previous **AWT**, URS adds agents to the working agent group according to the formula, $1/4*(N-M)$, where **N** equals the number of selected agents and **M** equals the ideal set of agents.
- If the service factor is fine but **SLWarn** is less than the percentage that should be achieved in $3/4*Y$ seconds for X percent set as the service factor, then URS initiates one of the following preventive actions:
 - If the current **AWT** is greater than or equal to the previous **AWT**, URS adds agents to the working agent group according to the formula, $1/8*(N-M)$.

- If the current AWT is less than the previous AWT, URS tries to reduce the number of agents in the current working agent group according to the formula, $1/8*(N-M)$ for only those agents who are not ideal. (URS cannot reduce the number of working agents if all agents are currently handling interactions.)
- If the service factor, $SLWarn \geq Y$, and current AWT is less than the previous AWT, URS tries to reduce the number of agents in the current working group according to the formula, $1/4*(N-M)$ for only those agents who are not ideal. (URS cannot reduce the number of working agents if all agents are currently handling interactions.)

Default Destination, Prediction, and Timeout

The Service Level, Statistics, and Workforce Routing objects give the option of entering Prediction, Default Destination, and Timeout information in the Busy tab (see [Figure 176](#)).



Figure 176: Busy Tab, Service Level Routing Object

Default Destination

Routing objects can contain a **Default destination** (see Figure 176 on [page 343](#)) that temporarily overrides the default destination specified in the configuration option `default_destination` or by the default DNs at the route point. If the timeout is over and the interaction has not been routed, the interaction goes to the destination specified here, rather than to the red port. The interaction is then considered successfully routed.

To specify an optional **Default destination** on the **Busy** tab to use after timeout:

1. Select Distribute to destination.
 2. Select a Switch, Type, and Number for the destination from the drop-down list.

The destination types available are ACDQueue and routing point only.

Prediction Threshold

Prediction Threshold (see Figure 176 on [page 343](#)) predicts the possibility that the next interaction will be routed successfully. In the **Prediction** box, there is a **Use Threshold** check box. When this box is checked, you can enter a percentage. This number represents the percentage of interactions that have must have been successfully routed in the past in order to predict successful routing in the future. URS uses this percentage as follows:

Assume you enter a 70% Prediction Threshold. When you load the strategy with the routing rule and have a Prediction Threshold set, URS sets up a counter and assumes that 128 calls were routed successfully:

Each time an interaction goes to the red port in the routing strategy (because a target was not available), a “success” position (1) becomes an “unsuccessful” position (0). For example, after the first interaction goes to the red port, the counter appears as:

Next, assume 90 interactions have been routed, but 39 have gone to the red port. At this point, the prediction is less than 70% that the next interaction will be successfully routed.

Setting a value for the threshold is an optional way to fine-tune a strategy. If the prediction level drops below the specified threshold, the next interaction does not wait for a target, but is sent directly to the red port. It is treated as though it waited but there was no ready agent. For example, if you specify a threshold of 60 percent but the prediction is 59 percent, the interaction will not wait for a target.

After the routing rule has been skipped 10 times, meaning that 10 interactions were sent to the red port because the threshold was exceeded, the prediction is changed one percentage point. In other words, 10 skipped interactions equals one successful interaction as far as prediction is concerned. This prevents the

interactions from going to the red port. When the prediction surpasses the threshold again, the next interaction is allowed to wait for a target.

Timeout

Timeout (see Figure 176 on [page 343](#)) specifies the amount of time, in seconds, URS is given to make a routing decision. At the end of this period, the interaction goes to the red port or to the optional Default destination if specified in the routing rule.

You can specify a destination for interactions after timeout in two ways:

- Select a Default destination in the properties dialog box (see Figure 176 on [page 343](#)) for the routing rule. See “Default Destination” on [page 344](#) for more information.
- Connect the Error segmentation object to the red port. From the Function drop-down list, select ApplyBusinessRule. From the Error drop-down list, select -001 Timeout. Then connect a Routing object to the output port of the Error object.

If you do not specify a destination for interactions after timeout, the interactions are automatically routed to the default destination.

In a Service Level routing rule, URS checks the state of the routing rule every 10 seconds, so the timeout is rounded up to the nearest 10 seconds. If the timeout is set to zero, URS actually waits 10 seconds. Similarly, if the timeout is set to 15 seconds, the rule does not timeout until 20 seconds.

To fulfill a Service Level routing rule, the timeout should be greater than or equal to the specified time in the service factor parameter.

Interaction Priority and Service Level Routing

The routing of an interaction is not only affected by the set of agents that is available but also by the interaction’s priority. URS also calculate this.

The initial priority of an interaction at the start of a strategy is based on the Service Factor. Every 5 seconds (previously 10 seconds in 6.5.x), as the interaction waits for the next available agent, URS checks the timeout value for the routing rule. If the interaction has waited longer than the timeout, the interaction is default-routed. If not, the interaction continues waiting. Every 5 seconds (previously 30 seconds in 6.5.x), interaction priority increases. URS expands the agent group, if necessary, to maintain the service factor.

Priority is calculated based on the Service Factor values. If the criteria is X percent of interactions in Y seconds, the initial priority value is calculated using the formula $X*2/Y$. After 5 seconds (previously 30 seconds in 6.5.x), each successive calculation is based on the formula:

`Current priority level + X * 5/Y.`



Use this object to route voice interactions. It uses a routing rule so that URS can obtain the values of defined statistics for targets from Stat Server (see Figure 177).

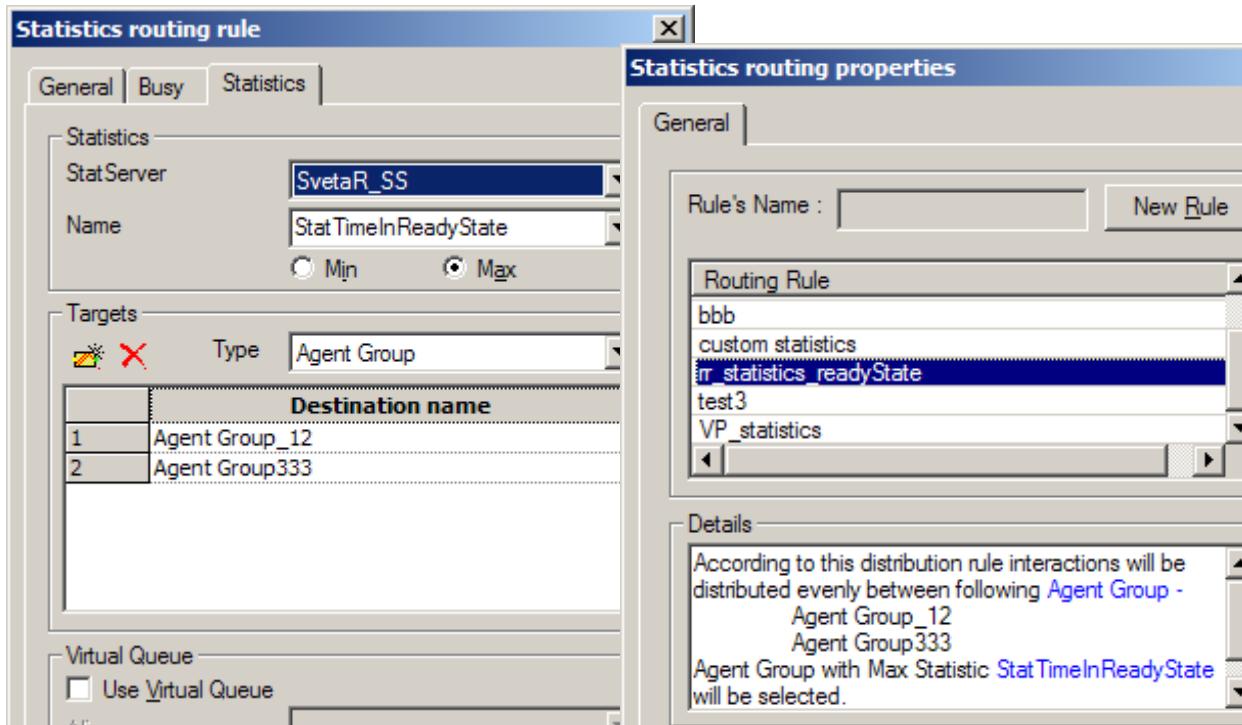


Figure 177: Statistics Routing Rule and Object Properties Dialog Box

Warning! If configuring Skills or Agent level routing and your site has a large number of agents, Genesys recommends using the `StatAgentLoading` statistic to select agents. Using the deprecated `StatTimeInReadyState` statistic can result in high CPU usage even if there are no calls.

Depending on whether the Max or Min option is selected under **Statistics** (see Figure 177), the available target with the maximal or the minimal value of the statistic receives the interaction.

Note: See the “[Routing Statistics](#)” chapter for information about predefined statistics, which can be selected in the Statistics area of the dialog box.

In the **Targets** section of the dialog box used for defining the routing rule:

- From the Type drop-down menu (see [Figure 177](#)), select a type (ACDQueue, Agent, Agent Group, Destination Label, Place, Place Group, Queue Group, Routing Point, or Skill).
- Add the number of targets desired using the Add Item button.
- For each target, from the Destination Name column, click the arrow to display a dialog box.

For Agent, the value is the employee ID. For Agent Group, Place, and Place Group, the value is the name of the corresponding object in Configuration Layer. For Routing Point, ACDQueue, and Destination Label, the value is the Alias of the corresponding DN in Configuration Layer. (The target type in this dialog box is preselected and no other type can be selected in this dialog box.)

For Skill type, create an expression in the Expression Builder dialog box.

See “User-Defined Statistics” on [page 732](#) for a list of recommended statistics that can be defined and used. For detailed explanations of all the relevant parameters, refer to the Genesys Stat Server documentation.



Switch-to-Strategy

Use the Switch-to-Strategy object for voice interactions. This object uses a routing rule to tell URS to route the interaction to the starting point of another strategy (see [Figure 178](#)).

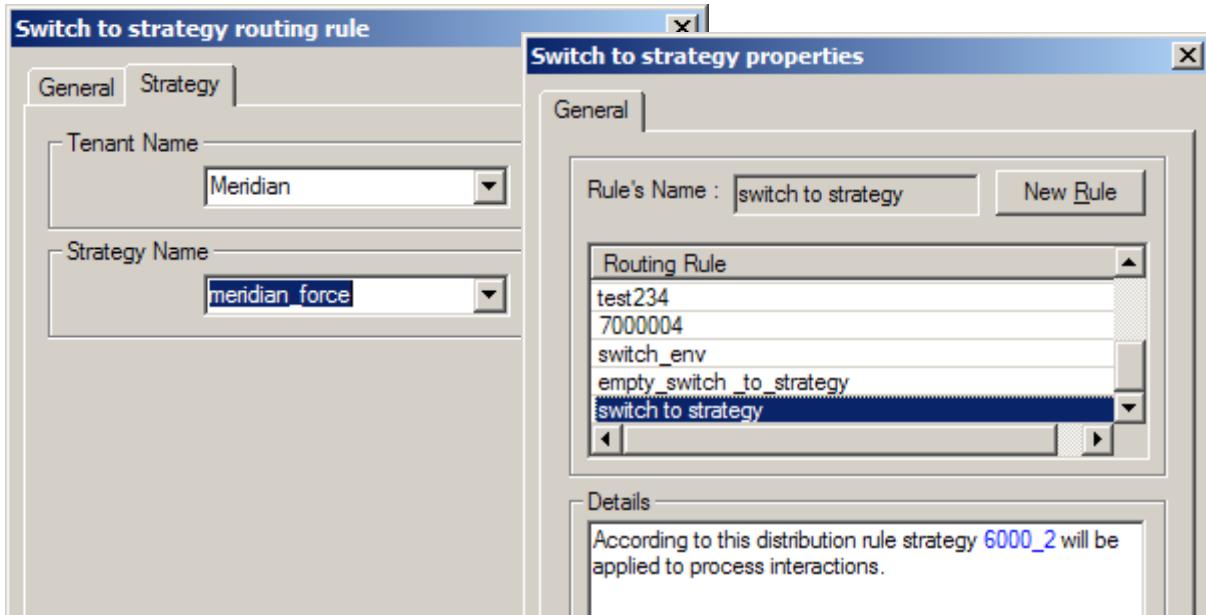


Figure 178: Switch-to-Strategy Properties Dialog Box

The interaction cannot return to the original strategy. All local variables in a new strategy that have both the same type and name as the original variable are initialized with the value of the original variable.

This object enables you to switch to a strategy from another tenant. In order to use this functionality:

- You must have access to the tenant's environment. The original strategy and routing rule should be created under Environment/Tenant in Configuration Manager.
- All tenants must be on URS's tenant list (Tenants tab of the URS Application object).
- The URS `change_tenant` option must be set to `true`.

Warning! Be careful that you do not design the new strategy to which the object is switching such that the new strategy calls the original strategy in which the Switch-to-Strategy object is located, thus causing an infinite loop.



Use the Workbin object (applicable to all non-voice media types) to route an interaction to a storage area associated with an Agent, Agent Group, Place or Place Group. For example, a Supervisor Workbin may hold draft e-mail responses. Workbins are created in the Interaction Design window (see [Figure 179](#)).

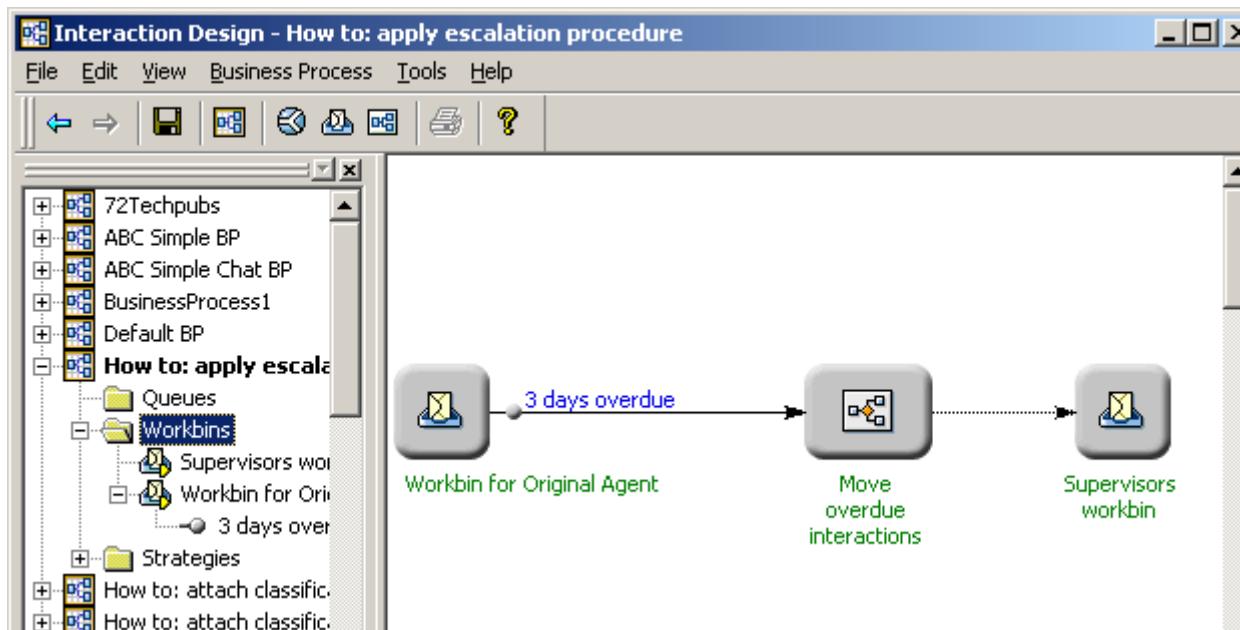


Figure 179: Workbin in the Interaction Design Window

Once the workbin is created in the **Interaction Design** window (see [page 98](#)), it appears for selection in the Workbin object.

Note: For information on creating workbins, see the *Universal Routing 7.6 Business Process User's Guide*.

Workbin General Tab

[Figure 180](#) shows the General tab of the **Workbin Properties** dialog box.

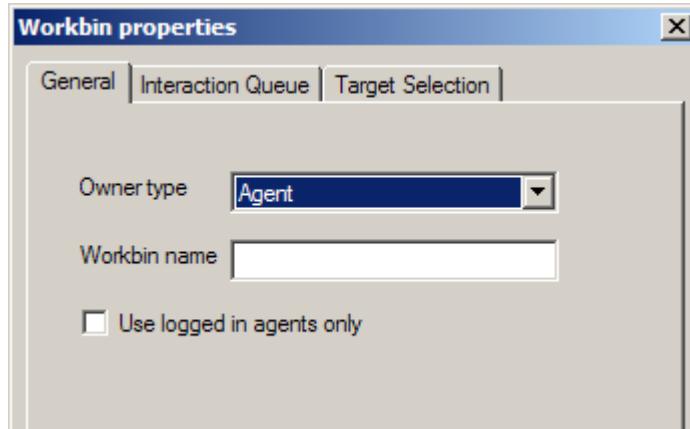


Figure 180: Workbin Properties Dialog Box, General Tab

Use the information in [Table 109](#) to complete the General tab.

Table 109: Workbin Object, General Tab

Parameter	Description
Owner type	<p>Click the down arrow and select one of the following: Agent, Agent Group, Place, Place Group. The Workbin name field uses your selection to determine the type of workbins to display for selection.</p> <ul style="list-style-type: none"> If you change the type of workbin Owner in the Workbin object properties dialog box in the Interaction Design window, you will also need to change it in the Routing Design window in the strategy containing the Workbin routing object. Otherwise, no message will be generated in the IRD Details tab or in the Log tab of the Interaction Design window. This is because Check Integrity was not designed to distinguish the wrong types of existing objects as the case of selecting the wrong type of Owner in the General tab of the IRD Workbin properties dialog box.
Workbin name	Click inside the field and select a workbin previously defined in the Interaction Design window. Workbins are organized by business process.
Use logged in agents only	Select to have URS check the agent's status and only route to an agent who is currently logged in.

Workbin Interaction Queue Tab

Figure 181 shows the Interaction Queue tab of the Workbin Properties dialog box.

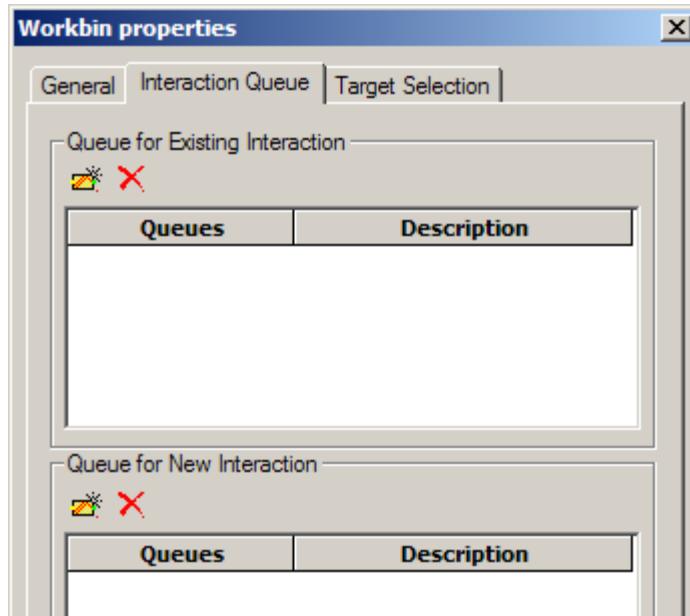


Figure 181: Workbin Properties Dialog Box, General Tab

Use the information in Table 110 to complete the optional Interaction Queue tab:

Table 110: Workbin Object, Interaction Queue Tab

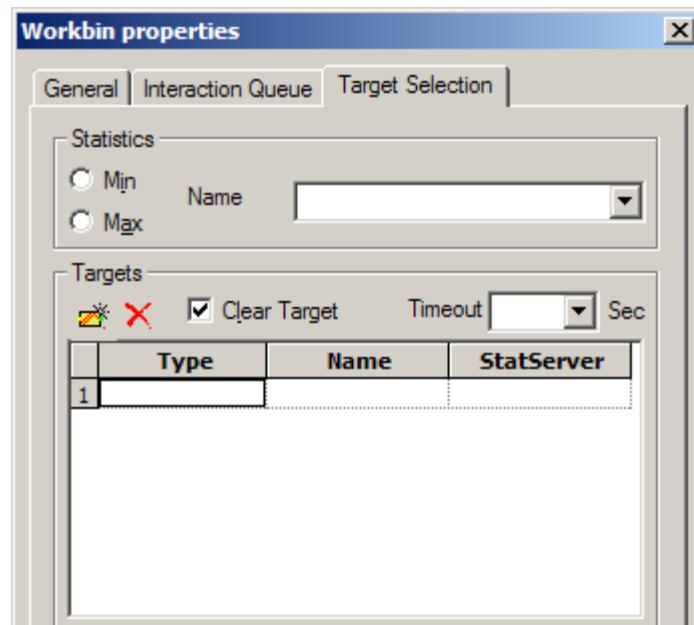
Parameter	Description
Queue for Existing Interaction	Click the button to add a new entry ().
	Click in the empty Queues text box and a down arrow appears. Click the down arrow and select an interaction input queue previously defined in the Interaction Design window (see page 98). The Stop Processing value is predefined. This queue will be used for the existing interaction after the agent has retrieved the interaction from the workbin and is through working with the interaction.
Description	Click in the empty text box and a down arrow appears. Click the down arrow and enter or select a description, which will be used as a hint on the agent desktop. For example, after the agent constructs a response, the response may need to be checked by QA. In this case, the description would prompt the agent to place the interaction in the queue read by QA.

Table 110: Workbin Object, Interaction Queue Tab

Parameter	Description
	If there is more than one queue, repeat Steps 1~3.
New Interaction	Repeat the above process to define interaction output queues when a new interaction is created. This queue will be used for any new interaction the agent creates as a result of retrieving the existing interaction from the workbin and is through working with it.

Workbin Target Selection Tab

Figure 182 shows the Target Selection tab of the Workbin Properties dialog box.

**Figure 182: Workbin Properties Dialog Box, Target Selection Tab**

Use the information in [Table 111](#) to complete the Target Selection tab:

Table 111: Workbin Object, Target Selection Tab

Parameter	Description
Statistics	<p>The Statistics area is enabled after you click under Targets.</p> <p>Optional. If more than one target is available from the target list when an interaction encounters the Workbin object, URS uses the Statistics information. Also see “List of Predefined Statistics” on page 718.</p> <p>Note: When routing to a target that is a meta-object (such as a Group of Agents or a Group of Places) and using any Routing object statistic except StatAgentLoading and StatAgentLoadingMedia, URS does not expand the target to its underlying level (for example, the underlying Agents belonging to the Group). Instead, URS applies the statistic directly to the meta-object.</p>
Min Max	<p>Click the Min or Max button to indicate whether URS should use the minimum or maximum value of the statistic to be selected next. See page 332 for information about the StatAgentLoadingMedia statistic.</p>
Name	<p>Click the down arrow and select the statistic to be used when more than one target is available.</p>
Targets	<p>Click the button to add a new entry () to enable fields. The strategy will not compile unless a target is specified.</p>
Type	<p>Click in the empty Type text box and a down arrow appears. Click the down arrow and select the target type (Agent, Agent Group, Variable, or Skill).</p>
Name	<p>Click in the empty text box and a down arrow appears. Click the down arrow and select the specific Agent, Agent Group, Variable, or Skill. If you selected Skill as the target type, the Skill Expression Properties dialog box opens where you can construct a skill expression (see Figure 40 on page 89). See “Logical Expressions” on page 87 for help in constructing a skill expression.</p>
Stat Server	<p>Click in the empty Stat Server field and a down arrow appears. Click the down arrow and the Key dialog box appears. In the Key dialog box, click the Value down arrow and select the name of the Stat Server to supply statistical information used for routing.</p>
	<p>Optional. If URS should consider more than one target after the Timeout number of seconds, repeat the above steps.</p>
Clear Target	<p>Optional. Click the check box if targets listed in the object should be retained after the interaction moves on through the strategy and encounters other Workbin object.</p>

Table 111: Workbin Object, Target Selection Tab (Continued)

Parameter	Description
Timeout Sec	Optional. Enter the timeout number of seconds to specify the time an interaction waits for an available target. If the timeout expires, the interaction is routed to the red port.
Use Virtual Queue	Set a Virtual Queue for by specifying the Alias, Switch, and Number. See important note on page 333 .



Workforce

Note: A Share Agent by Service Level Agreement solution, as described in the *Universal Routing 8.0 Routing Application Configuration Guide*, is incompatible with the use of a Workforce routing rule.

The Workforce object uses a routing rule so URS can communicate with Genesys Workforce Management (WFM) components to route an interaction based on agent schedules and activities. In IRD, you first create Workforce routing rules. After naming and describing the rule in the General tab, you complete Workforce and Busy tabs (see [Figure 183](#)).

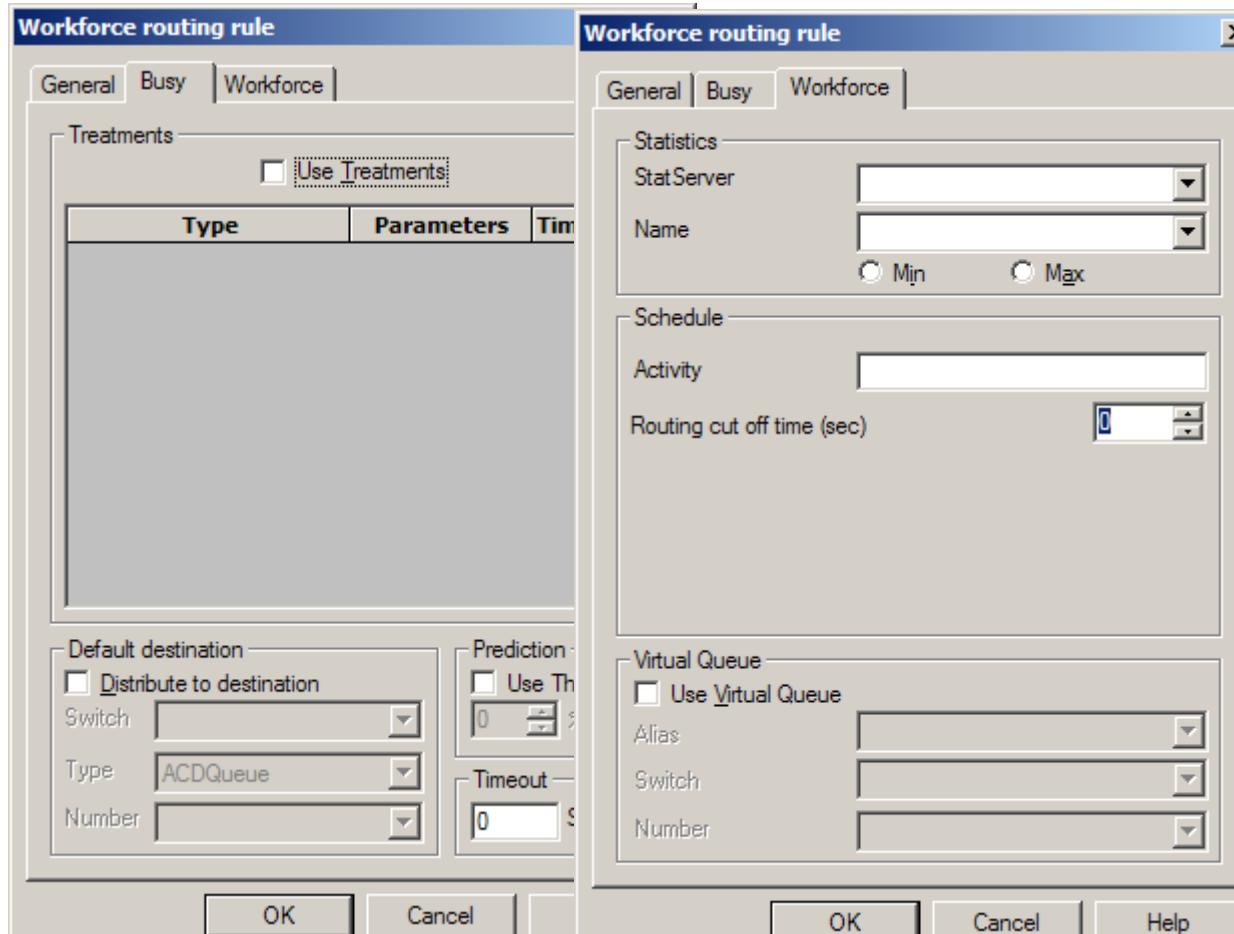


Figure 183: Workforce Properties, Busy and Workforce Tabs

Workforce Routing Rule Workforce Tab

In the **Workforce** tab, you enter:

- A **Stat Server** and **statistic Name**—URS will use these to determine which agent to route the interaction to if more than one agent scheduled for that activity is available. Select a **Max** or **Min** option to specify whether the maximum value or minimum value of the statistic should be used to determine which of the available targets should receive the interaction.
- An **Activity** for the rule—The name of the activity by which URS will determine agent availability based on the schedules and activities created in the WFM Configuration Utility and stored in the WFM database. The activity name **must** be entered exactly as it appears in the database. See the *Workforce Management 8.1 Configuration Utility Help* for information on creating activities.

Note: Because Workforce Management Activities data is not integrated into the Configuration Layer, URS cannot take advantage of any dynamic notification when changes are made to Activity names. You must manually enter Activity name changes.

- **Routing cut off time**—the period that URS stops sending interactions to an agent for the activity. When deciding on the cutoff time, consider the duration for a typical interaction and the threshold that an agent can allow an interaction to overlap into the time of the next scheduled activity. For example, if a typical interaction lasts 3 minutes and the threshold is 2 minutes, the cutoff time might be the difference between them or 1 minute. URS would no longer send interactions specific to that activity to a particular agent when only 1 minute remains for the agent to spend doing that activity. This cutoff time depends on the handle time and the flexibility of the activity or nonactivity (break) that follows in the agent's schedule. Handle time is the interval from the moment an agent first responds to a call to the moment the agent is ready to take another call.

Workforce Busy Tab

In the **Busy** tab, enter treatment information (see “Busy Treatments in Routing Objects” on [page 366](#)), a default destination, and prediction information (see “Default Destination, Prediction, and Timeout” on [page 343](#)). You also enter a timeout:

- **Timeout**—the period that URS tries to route an interaction for this routing rule. When the timeout expires, URS looks to the object in the strategy to determine where to route the interaction (see “Busy Treatment Time-Outs” on [page 367](#) for more information).

Selecting a Workforce Routing Rule

After configuring Workforce routing rules, you place a Workforce object in a strategy and select which Workforce routing rule to use for this object.

[Figure 184](#) shows an example dialog box for selecting a Workforce routing rule.

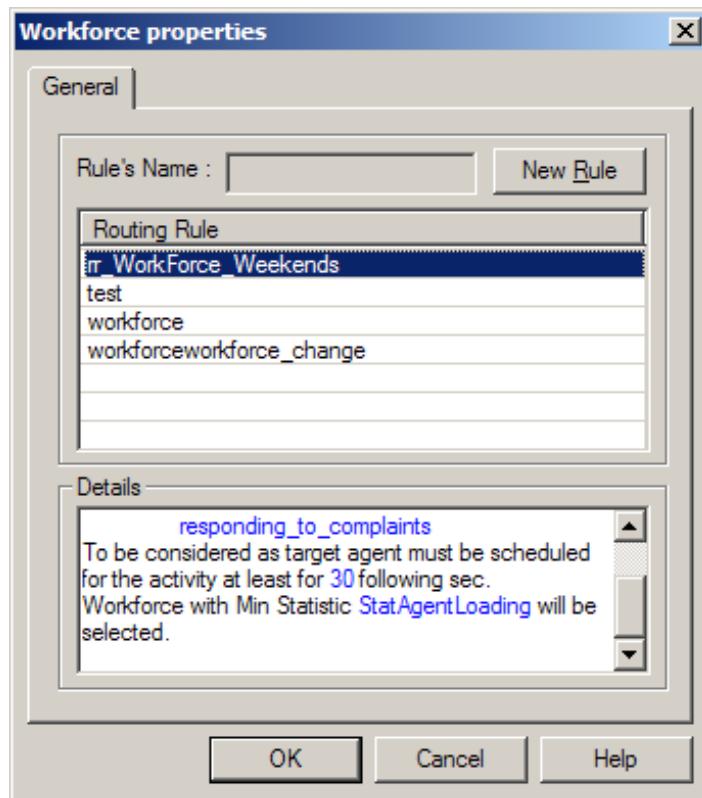


Figure 184: Workforce Properties Dialog Box

Configuring Workforce Management

URS uses the `http_port` option to communicate with WFM. See the *Universal Routing 8.1 Deployment Guide* for configuration information on this option.

After you configure Universal Routing to work with WFM, URS scans schedules and activities stored in the WFM database as described below.

Every 15 minutes, URS scans and analyzes the schedules and activities stored in the WFM database. Based on this analysis, URS routes an interaction to an available agent scheduled for a particular activity during this interval using the information specified in the Workforce routing rule for the Workforce object.

Note: Because the Workforce object uses agent schedules and activities to route interactions, workforce routing is an advanced form of skills-based routing. This means that interactions are directly routed to the qualified agent. If workforce routing is used at the network level, agent-level routing at the IN network level must first be set up and functional.

Warning! If an agent schedule changes or another agent schedule is added after URS scans the database at the end of one interval but before the next scan, the changes are not taken into account when routing to an agent. These changes are recognized the next time URS scans the database.

Error Codes for Routing Objects

With the exception of the Force object, the following error codes might appear for all other Routing objects:

- `-001 Timeout`
- `0001 Unknown error`
- `0008 Routing done`
- `0013 Remote error`
- `0018 Unknown object`
- `0019 Translation failed`

Postrouting Following a Routing Object

The object following a Routing object, and all the objects after it, compose the sequence of postrouting actions. These objects will only be executed if the interaction is successfully routed to one of the targets specified under the Selection object.

Virtual Queues in Routing Objects

A virtual queue is not a physical queue like an ACD in a PBX switch. A virtual queue is a logical queue (as opposed to a physical queue) where an interaction is queued when the required targets are not available. Virtual queues in Routing objects are used for reporting purposes and also statistic collection for URS. Virtual queues are recommended for any routing target other than ACDQueue, Route Point, Destination Label, or Queue Group.

One way of using a virtual queue is as follows: An interaction comes in to a routing point and URS routes the interaction to an IVR while waiting for an agent; the interaction is established at the IVR port and is no longer shown in queue. However, the interaction must still be routed to an agent. A virtual queue can be used by URS to show that this interaction is still in queue for an agent. This information is also used when reporting statistical information about this particular interaction.

Virtual queues can be specified as a variable or string for the following:

- Route Interaction (see [page 322](#)), Workbin (see [page 348](#)), and Selection objects (see [page 326](#))

- If object (CallsDistributed, CallsEntered, DistributedPercentage, DistributedWaitingTime, InVQWaitTime, NotDistributedPercentage, NotDistributedWaitingTime, and SelectDN functions) (see [page 129](#))
- Assign object (CallsDistributed, CallsEntered, DistributedPercentage, DistributedWaitingTime, InVQWaitTime, NotDistributedPercentage, NotDistributedWaitingTime, and Select DN functions) (see [page 120](#))
- Function object (CallsDistributed, CallsEntered, ClearTargets, SetVQPriority, DistributedPercentage, DistributedWaitingTime, InVQWaitTime, NotDistributedPercentage, NotDistributedWaitingTime, and Select DN functions) (see [page 127](#))

Note: For all functions with the Virtual Queue parameter, select the value type (variable or virtual queue) and the value (variable name or virtual queue name).

For the Route Interaction, Selection, and Workbin objects, when you select the Select Virtual Queue from the drop-down menu, specify the alias, switch (name), and number for the virtual queue. The alias can be a string or a variable.

Support for Virtual Queues Reporting

To address reporting requirements, URS attaches different types of data to interactions.(see the report_targets option on [page 660](#)). Part of this reporting-attached data is used to facilitate virtual queues reporting. This virtual queue-attached data is not controlled by any option and is always attached. It consists of the following keys:

- RVQID—The `VirtualQueue` ID of the virtual queue to which the interaction was distributed. It is attached immediately before the interaction is routed to its destination target.
- RPVQID—The `VirtualQueue` ID of virtual queue into which the call is placed. It is attached when the `EventQueued` event is distributed for the call for the virtual queue. In addition, at the beginning of strategy execution URS looks for any data for the interaction that might be attached with this key. If attached data is found, URS deletes it.

The `VirtualQueue` ID is a special value (GUID) that uniquely identifies the entrance of call into certain virtual queues. It is created by URS when the call enters into the virtual queue and is present in all `VirtualQueue` events that URS distributes.

Statistical Objects (Target Types)

In the process of running strategies, URS must request Stat Server about different statistics for different objects. These objects are referred as *statistical*

objects or *targets* (since most common use of statistical objects is to specify routing targets).

However, statistical objects can refer to much more than routing targets. For example, URS can request statistics for virtual queues or campaigns, which are statistical objects but not routing targets. In this case, the word *target* is used in the broader meaning of *statistical object* and does not specifically indicate a routing target.

Also use of a statistical object does not always involve Stat Server. For example, Stat Server is not required to send call to ACDQueues or Destination Labels. Target types include an individual agent (**Person**), an agent **Place**, an **Agent Group** defined by some criteria, an agent group defined by location and known as a **Place Group**, a **Queue Group**, a **Routing Point**, a **Destination Label**, an **ACD Queue**, a **Campaign**, or a non-configured (not-monitored) DN (see Figure 174 on [page 329](#)).

These objects have unique names that are configured in Genesys Configuration Manager. See [Table 112](#) for a description of these targets.

Table 112: Target Types

Target Type	Description	Symbolic Designation
ACDQueue	A point in the PBX or ACD where interactions wait for an available target. This type also includes virtual queues. ^a	Q
Agent	An individual that handles interactions and can move about the contact center to occupy different places.	A
Agent Group	A group defined by shared characteristics such as skill.	GA
Campaign	A campaign is a flexible master plan that organizes calling lists for dialing calls and handling call results. Campaigns are created in Configuration Manager or Outbound Contact Wizard.	C
Campaign Group	A campaign can be assigned to multiple campaign groups; however, a campaign group can participate in only one campaign in Running status at a time. Note: See “Campaign Group Targets” on page 362	GC
Destination Label	A label, defined on a network switch and controlled by Network T-Servers, that enables interactions to be routed to a remote destination.	DL

Table 112: Target Types (Continued)

Target Type	Description	Symbolic Designation
Interaction Queue	A queue used for non-voice interactions in a business process (see page 149) executed by Interaction Server. You might also call this type of queue a persistent queue.	
Non-configured (non-monitored) DN	A DN that is not configured in the Configuration Database and that, therefore, is not monitored by T-Server. For more information, see “DN Target Type” on page 589 .	DN
Place	A specific station with one or two associated DNs.	AP
Place Group	A group of stations defined by common location.	GP
Queue Group	<p>A group of queues (DNs) can function as a target.</p> <p>Warning! Universal Routing 7.2 through 7.6 does not support Routing objects in strategies that use routing rules with targets of type Queue Group.</p>	GQ
Routing Point	A logical point in the PBX from where interactions are routed (CCT, CDN, VDN). This also includes Virtual Routing Points. ^b	RP
Skill	<p>See “Skill Target in Routing Objects” on page 363.</p> <p>Note: Although IRD presents Skill as separate target type, URS considers it as an agent group type target. Therefore, IRD silently converts it to the agent group target type.</p>	Not applicable
Variable	<p>A variable that represents one or more targets of listed above types. If you are using more than one target, enter them as a comma-separated list, with the Stat Server specified as a part of the target name.</p> <p>See also “Variables in Objects and Expressions” on page 92.</p>	Not applicable

^a Use a routing queue for switches with no routing points so they function like a routing point.

^b Use virtual routing points for handling media-type interactions such as e-mail, chat, and so on.

Note: In case of a workbin(see [page 348](#)), the owner of workbin (agent or agent group for example) is reported in the attached data as a target rather than the workbin itself.

Target Properties

After you select a target Type (see Figure 174 on [page 329](#)), you define target properties. Target properties define the interaction recipient by the following characteristics:

- Type
- ID—configured name or identification associated with the target in Configuration Layer. In Figure 174 on [page 329](#), the column is Name.
 - For a Routing Point, Destination Label, or an ACDQueue, the ID is the Alias of the corresponding DN in Configuration Layer.
 - For an Agent, the ID is the Employee ID of the corresponding Person in Configuration Layer.
 - For an Agent Place, an Agent Group, Campaign, Campaign Group, or a Place Group, the ID is the Name of the corresponding object in Configuration Layer.
- Location—The name of the Genesys Stat Server (see Figure 174 on [page 329](#)) that will report on the target, as defined in Configuration Layer. This location must be defined if it is different from the name given in the URS configuration option default_stat_server (see [page 639](#)).

Formats for Specifying Targets

A target in a strategy is specified as Name@statserver.type. If any of the target components (name or Stat Server or type) are different, URS considers them to be different targets. Stated another way, two targets with the same agent defined with different Stat Servers are not the same targets.

Every target must have three properties: name, location, and type.

Even if URS does not use Stat Server to identify some types of targets (for example, targets of type DL or DN), location is still mandatory part of the target specification. If you do not specify a location or a type of target, URS retrieves them from those set in the URS Application object.

Note: For a non-configured DN, the ID is a string having the format <phone number>@<switch>. However, the <switch> parameter can be omitted.

Important Information on Targets

- Target names are limited to alphanumeric characters, the at (@) symbol and dot (.) unless the full format style, `TargetName@StatServerName.TargetType`, is specified.
- Two or more Stat Servers cannot be used to monitor the same target. Since targets are specified in the following format, `TargetName@StatServerName.TargetType`, each target has a unique target description (including the Stat Server name). If two Stat Servers (`StatServer_A`, `StatServer_B`) monitor the same agent target for example, there would be two different target descriptions, `AgentA@StatServer_A.A` and `AgentA@StatServer_B.A`. These different target descriptions would be understood as different agents.
- In the case of forced routing (if the function Force has been used in the strategy before the Routing object in which the target is placed), the ID must be a DN configured, and the location must be the name of the switch where the DN is located; the Type is disregarded.
- The alias automatically generated for routing queues and virtual routing points do not make them automatically identifiable. Since routing queues are included in the ACDQueue and Routing Point target types and virtual routing points are included in the Routing Point target type in IRD, one way to easily identify one target type from another is by creating a unique alias for each type. For example, when configuring a routing queue, you could specify an alias name that includes the prefix or suffix RQ.
- The name of any single target (including skill groups) cannot exceed 254 bytes.

Campaign Group Targets

You can use this target type in the Selection and Route Interaction objects, and in the SelectDN and SData functions. You cannot use this target type with routing rules. The Type notation for campaign groups is GC; example:
`CampGroup@statserver.GC`

Every Campaign Group references some Agent or Place Group. With one important exception, Campaign Group targets behave identically to targets associated with Campaign Group Agent/Place Groups. The exception is as follows:

URS considers Campaign Group targets valid if, **and only if**, the associated Agent/Place has the value of statistic `CurrentAgentAssignment` matched to the Campaign Group.

When you use Campaign Group as a target, this results in URS opening the statistic `CurrentAgentAssignment` for every participating Agent/Place. Opening of this statistic for every participating Agent/Place affects not only outbound routing (if any), but in addition:

If this statistic is open and has a value matched to some Campaign Group (= not inbound), the interaction cannot be routed to an Agent/Place that is part of any other group.

Agents/Places that do not have the CurrentAgentAssignment statistic open (meaning they are not included as part of any Campaign Group addressed by URS so the statistic does not have to be opened) are considered as valid targets for any non-outbound activity. Use the environment option (see [page 639](#)) to handle this situation.

If the same URS is not handling the corresponding outbound interactions, set the URS option environment to outbound. Once this option is specified as outbound, URS opens the CurrentAgentAssignment statistic for every Agent/Place it works with.

Note: URS supports proactive customer contact and multi-Campaign agent management (see “Calling List Overview” on [page 295](#)). To enable routing of outbound calls/interactions to Campaign Groups, the URS installation package includes a prewritten strategy design to handle outbound calls as well as sample strategies. For more information, see the *Universal Routing 8.1 Deployment Guide*.

Deleting Unused Target Objects

URS may delete agent groups/place groups content statistics if they are not used long enough.

Group content statistics is a candidate for deletion if:

- There is no skill content statistics which depends on this group content statistics.
- The group content statistic is not accessed for four days.

URS checks if it can delete content statistics when updating events from the Configuration Server for this group, or any agent/place/skill.

Skill Target in Routing Objects

IRD has four Routing objects that enable to use Skill type routing targets:

- Route Interaction (see [page 322](#))
- Selection (see [page 326](#))
- Statistics (see [page 346](#))
- Workbin (see [page 348](#))

[Table 113](#) compares the four objects.

Table 113: Comparison of the Route Interaction, Selection, Statistic, and Workbin Objects

	Routing	Timeout	Statistics
Route Interaction, Selection, Statistic, and Workbin	Route to skill	Specify timeout	Specify statistic
Route Interaction, Selection, and Workbin	Can route to a variable	Can keep targets available using the clear target option	
Statistics	Uses the Statistics routing rule, which provides reuse and dynamic strategy changes		

Table 114 on [page 364](#) highlights differences and similarities of the objects.

Table 114: Differences among the Route Interaction, Selection, Statistics, and Workbin Objects

	Features
Selection and Statistics	<ul style="list-style-type: none"> Routes to ACD Queue, Agent, Agent Group, Destination Label, Place, Place Group, Routing Point, or Skill. Allows time-outs (the number of seconds that a customer interaction will wait for a target to become available). Allows statistics that decide which agent is chosen if multiple agents are available, such as the agent that has been ready the longest.
Route Interaction	<ul style="list-style-type: none"> Routes to Agent, Agent Group, Campaign Group, Place, Place Group, or Skill. Routes to variables. This can be used with IRD functions or database lookups that return the targets in a variable. Allows statistics, such as <code>StatAgentLoadingMedia</code>, that decide which agent is chosen if multiple agents are available.
Selection	<ul style="list-style-type: none"> Routes to variables. This can be used with IRD functions that or database lookups that return the targets in a variable. Allows targets to be retained for an interaction if there are multiple Selection objects and the Clear Targets check box is not selected.

Table 114: Differences among the Route Interaction, Selection, Statistics, and Workbin Objects (Continued)

	Features
Statistics	<ul style="list-style-type: none"> Uses a Statistics routing rule. Routing rules allow the routing point decision to be specified once and used multiple times within a strategy or across strategies. Users can change the target specifications in routing rules. The change affects the strategy without reloading.
Workbin	<ul style="list-style-type: none"> Routes to Agent, Agent Group, or Skill. Routes based on content of a variable. This can be used with IRD functions or database lookups that return the targets in a variable. Allows statistics, such as <code>StatAgentLoadingMedia</code>.

When you specify the target type `Skill`, you select a Name and a Stat Server. The Name for the `Skill` type is actually a skill expression that you create in the `Skill Expression` dialog box (see Figure 40 on [page 89](#)). This dialog box is also accessed by functions that require a skill expression including `CreateSkillGroup` (see [page 514](#)) and `Multiskill` (see [page 526](#)).

Skill Targets and Skill Expressions

In the `Skill Expression` dialog box under Type (see Figure 40 on [page 89](#)), you specify the following:

- A data type: `Skill`, `Statistic`, or `Variables`.

Warning! Skill names are limited to alphanumeric characters and underscores. The name cannot begin with a digit. See [page 31](#) for important limitations regarding skill expressions.

- A data Name. What appears in this section depends on which data type you selected. For example, when you select the `Skill` data type, the Name field lists the skills defined in Configuration Manager.
- A condition of the `Skill` (an arithmetic expression):
 - `!=` (not equal to the indicated level)
 - `<` (less than the indicated level)
 - `<=` (less than or equal to the indicated level)
 - `=` (equal to the indicated level)
 - `>` (greater than the indicated level)
 - `>=` (greater than or equal to the indicated level)

- A Value that can be a Skill Level, also defined in Configuration Manager or a variable.

If a variable is defined in the strategy and then is used as a comparable in a skill expression, the value of the variable will be interpreted as the skill name when the expression is evaluated. Otherwise, the string itself will be interpreted as the skill name. See “Variables in Objects and Expressions” on [page 92](#).

When the order of logical operations is not explicitly specified by parentheses, the operation to the left has precedence over the operation to the right. For example:

`Portuguese > 5 | Africa = 7 & SpTours >3`

is equivalent to

`(Portuguese > 5 | Africa = 7) & SpTours > 3`

Using a Skill Expression to Exclude an Agent

To exclude an agent from being selected during skills-based routing, you can take one of the following actions:

1. Delete the skill in the agent’s profile in Configuration Manager.
2. Set the agent’s skill equal to zero (`English = 0`).

Important Note on the Absence of a Skill

URS interprets the absence of a Skill as the agent having the Skill with a zero level (`English = 0`). For example:

- Assume the desired Skill is `English < 8`. If one agent has skill `English = 0` and another agent has no English skill at all, both agents can be selected. In this case, with `English < 8`, URS includes all agents with the English skill less than 8, and also agents with no English skill at all. If you want to include agents with any Skill between 1~7, then state `English >= 1` and `English < 8`.
- Assume the desired Skill is `English >= 0`. Again, both agents can be selected since URS interprets `English = 0` as the same as no English at all.
- Assume the desired Skill is `English > 0`. Neither of the above agents will be selected.

Busy Treatments in Routing Objects

This section contains information on the Busy tab found in the properties dialog box when defining Percentage, Service Level, Statistics, and Workforce routing rules (see [page 312](#)), as well as the Busy tab in the Routing Selection object (see [page 326](#)).

Note: Some switches do not support specific treatment calls. For a list of supported treatment functions, refer to the “Supported Functionality in T-Server” chapter in your Genesys T-Server documentation. For example, refer to the Supported Functionality in T-Server for Alcatel A4400 chapter in the *T-Server for Alcatel A4400 Reference Manual*.

Busy treatments are played to callers while they are waiting in a queue. Any number of treatments can be used for an interaction. The waiting interaction receives each treatment for the specified duration, in the specified order.

Once a Busy treatment is applied to an interaction, T-Server sends `EventTreatmentApplied` to URS. This event signals to URS that treatment has started and it can begin to wait for the end of the treatment in order to start the next busy treatment. Ending of treatment is signaled either by `EventTreatmentEnd` or by expiration of treatment time-out. If the interaction is still waiting when all the treatments have been played through once, the treatments start again at the top of the list.

The list of Busy treatments available within Routing objects are the same treatments provided for Mandatory treatments ([page 418](#)), with the addition of the newly introduced Exit treatment in 8.0 (see the appropriate bullet point in the section “Important Information on Busy Treatments in Routing Objects” on [page 369](#)).

Variables are replaced with their values in treatments parameter at the moment the busy treatments are defined, not when they are going to be played.

Any time that URS finds an available target for the call, the previously-started and currently-played busy treatment (and entire busy treatments chain) will be interrupted in favor of routing the call to a target. Target availability is checked on starting a treatment (`EventTreatmentApplied`), on updates from Stat Server, and usually every 2 seconds (value of this time interval is configurable by option `pulse_time` (see [page 652](#)).

Busy Treatment Time-Outs

In general, the amount of time of the Busy treatments should equal the amount of time specified to wait for the target in the timeout of the Routing object. Here’s why:

- If the total amount of time for all treatments is less than the target timeout, the treatments will start again from the beginning. This may or may not be acceptable, depending on the treatments used.
- If the total amount of time for all treatments is greater than the target timeout, then not all treatments will be applied. This may or may not be acceptable, depending on the treatments used.

For some treatments, such as Play Announcement, you must know the length of the treatment so you do not specify a timeout that will cut off the

treatment. If you specify a timeout greater than the length of the treatment, T-Server ignores the timeout.

At the end of target timeout, the interaction goes to the red port or to the optional default destination, if specified.

Setting Independent Time-outs

In some cases, you may wish to set a Busy treatment timeout that is independent of the target timeout. For example, you may wish to use the same Busy treatments for two or more Target Selection objects.

Assume you have three target selections connected to the red port, with the timeout for each target selection set at 10 seconds ($10 \times 3 = 30$). Assume also that you have two Busy treatments. The first one is 20 seconds and the second one is 10 seconds ($20 + 10 = 30$).

To make the Busy treatment timeout independent of the target timeout, you must set the time-outs for all Busy treatments in the first Target Selection object and not set any treatments in subsequent Target Selection objects.

In this case, when moving to the next Selection object, URS automatically reuses the current treatment

If there is another set of Busy treatments in the second target selection object, then IRD automatically resets Busy treatments when moving to the second target selection object. In this case, you do not need to use function `ResetBusyTreatments`.

The `ResetBusyTreatment` function clears all previously set busy treatments from URS memory.

[Figure 185](#) shows two strategy scenarios that produce the same outcome.

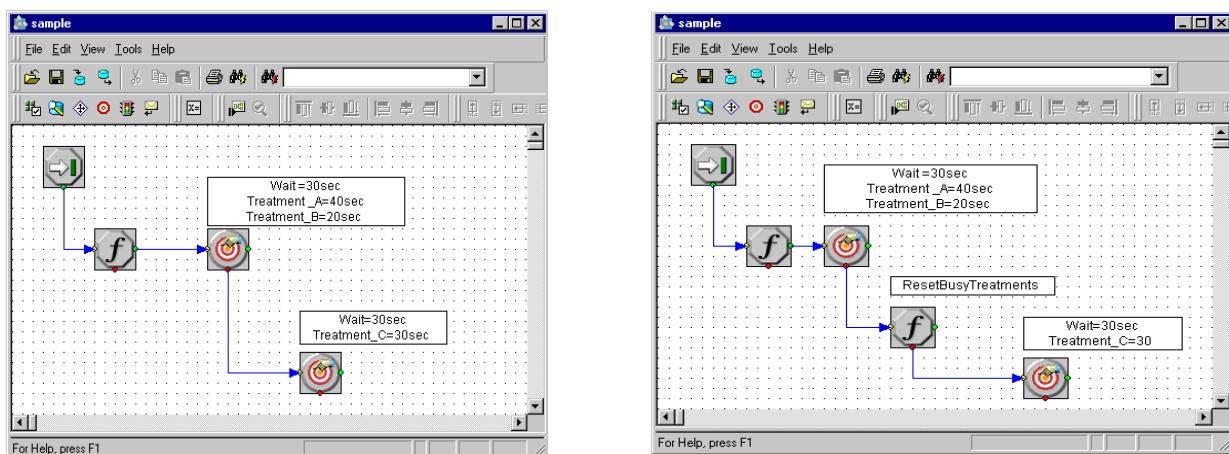


Figure 185: Strategies That Reset Busy Treatments

The outcomes of the first and second strategy are the same. IRD automatically resets the treatments in the first strategy. This accomplishes the same thing as `ResetBusyTreatments` in the second strategy.

Important Information on Busy Treatments in Routing Objects

- Use of extensive treatments might degrade URS performance since URS attempts to process an interaction after each treatment for every interaction. Therefore, always optimize the use of treatments when designing a strategy.
- Variables can be specified for a Busy treatment used in the Selection object only but not for objects that specify Busy treatments through a routing rule. Since variables are specific to a strategy, if you enter a variable into a Busy treatment for a routing rule, the variable will be interpreted as a string by other strategies that use that routing rule.
- If it is important for URS to adhere to call priority even for interactions that are sent to other DNs for Busy treatments, set the function KeepQueue to true. This will prevent URS from distributing interactions with a lower priority to the same target before the higher priority interaction is distributed. See Chapter 3, “Interaction Routing Designer Functions,” [page 423](#) or more information about the KeepQueue function.
- If a call is in a switch ACD queue, then the switch must handle the treatment since URS has relinquished control of the call.
- See the “[Starting Busy Treatments](#)” section below.
- While not a true Busy treatment, Exit is used to force immediate quitting (even if the specified waiting timeout has not yet expired) from the current target selection object, with reason Timeout.
- If you need to change current chain of busy treatments from the strategy, here are the only possible ways to do it:
 - Start a Mandatory Treatment (see [page 405](#)) or
 - Call ResetBusyTreatment function (see [page 530](#)) or
 - Use of a Routing object with a not-empty list of busy treatments.

Starting Busy Treatments

If URS cannot apply a busy treatment when the strategy requires it, it tries to reapply the treatment every half second until it is successful or until the treatment is canceled because the strategy requires that the interaction be moved into a different processing stage (for example, if the target becomes available).

Note: In previous versions, if URS could not apply a busy treatment because of the state of the call, it discarded this treatment. This sometimes meant that no busy treatments were played at all, and required special steps to be taken in the strategy to avoid such situations.

Note: Upon completion of a consult call transfer, URS runs a strategy on the condition that, if some busy treatment is played for this call, URS will reapply the currently-played treatment for the main call when the option `give_treatment` is set to `true`. The treatment will be applied with just the same parameters as for the original consult call. The chain of busy treatments will continue as usual after this treatment concludes.

If there is no currently played busy treatment (if option `give_treatment` set to `true`) the RingBack treatment is applied.

In previous versions of URS, if the option `give_treatment` was set to `true`, URS always played the RingBack treatment.

If the `give_treatment` option is set to `false`, no treatment will be applied (just as in previous releases).

URS cannot apply a busy treatment if, at the moment when the treatment should be applied, the call is in a state that does not allow the treatment to be applied. This can occur if the strategy contains sequential target Selection blocks and URS does not control the call transfer from an IVR port back to the routing point after an IVR Treatment defined in the first block has been applied. There are two valid scenarios known that could result in such a call state:

- URS does not monitor the IVR port where the call landed for a busy treatment and is not aware of the call state at the moment when the next treatment should be applied.
- URS monitors the IVR port, but the following race condition occurs: the previous treatment has been completed and the call is in the process of transferring back to the routing point (in other words, the call is on hold at the IVR port) at the moment when the next treatment should be applied.

If a call runs into one of these scenarios, failing to apply the subsequent busy treatment can lead to call default routing by the switch. In order to avoid such incidents, implement one of the following exception handling sequences in the strategy:

1. Precede the subsequent target Selection block with a `SuspendForTreatmentEnd[x]` instruction, where `x` is the number of seconds equal to the maximum duration of a treatment that may be in progress at this moment (see [page 615](#) for the `SuspendForTreatmentEnd` description). [Figure 186](#) shows an example strategy.

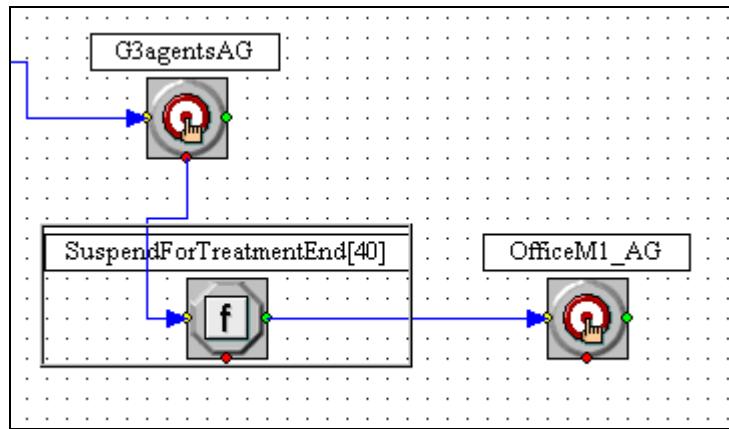


Figure 186: SuspendForTreatmentEnd

Note: The call will sit in the SuspendForTreatmentEnd[x] block and will not be routed to any targets that may become available until the treatment in progress is completed.

In the second scenario, where URS does monitor the IVR port, you might want to terminate the treatment in progress and start applying the new treatment chain as soon as the call comes out of the first target Selection block. In this case, consider the following modification:

2. Precede the subsequent target selection block with the following instructions, one after the other, as it is depicted in [Figure 187](#):

`DeliverCall['return:ok',DELIVERBACK]`

`SuspendForTreatmentEnd[x]`

If the DeliverCall instruction fails and the call exits the DeliverCall block through the red port, this indicates that the race condition occurred and the call is being transferred back to the routing point. In this case, the `SuspendForTreatmentEnd[1]` instruction should be executed before the subsequent treatment is applied to allow for the transfer to be completed.

[Figure 187](#) shows an example strategy.

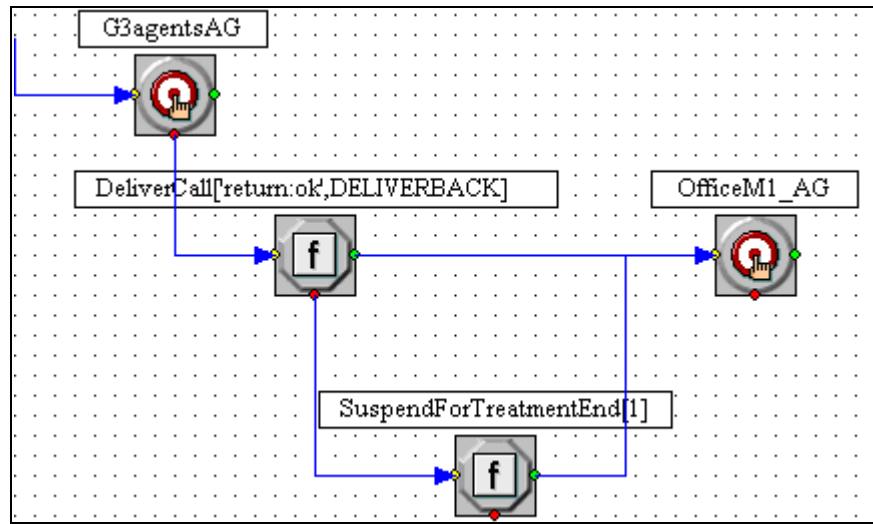


Figure 187: DeliverCall/SuspendForTreatmentEnd

Note: This method cannot be used for the first scenario, since URS is not able to execute the DeliverCall instruction if it does not monitor the IVR port where the call has landed.

Dynamic Busy Treatment at Selection Block

URS can unconditionally scan all treatment parameters (before sending the treatment request) for special escape sequences and replace them with corresponding values that enable different treatment parameters to be passed to the IVR application on every executed round of busy treatments.

The escape sequence is any fragment in the format [Variable Name]. The strategy must have a variable with [Variable Name] defined. The value of the [Variable Name] fragment is extracted and interpreted, and the resulting value is replaced by the value of the variable.

The strategy variable must be of type STRING and contain an expression as its value. The expression is similar to those created as threshold expressions in the target selection object. However, this expression is limited and must contain one of two functions only—`sdata[]` and `udata[]`.

Sample To add the value of the call-attached data with the `MyKey` key to a treatment parameter, the value of treatment parameter must be specified. For example, `{Param}`, where `{Param}` is the strategy variable of type STRING with the value set to `udata[MyKey]`.

Note: URS does not support the direct inclusion of an expression (rather than through a strategy variable) into treatment values.

Using Dynamic Busy Treatment

The following steps describe how to use the Dynamic Busy treatment:

1. Start IRD and enter the Configuration Server credentials. IRD will read configuration.
2. Click **New** to create a new strategy. IRD starts in the **Strategies** pane, by default.
3. Create two variables of type **STRING**. For example, `varStat1` and `varStat2`.
4. Assign values to these variables, as follows:
 - For `varStat1`, assign string
`'sdata[VQ1_vit_sw2.Q, StatExpectedWaitingTime]'`
 - For `varStat2`, assign string `'udata[MyAttachedData]'`
5. Add to the strategy by attaching the data block that attaches data 123 with the `MyAttachedData` key.
6. Add to the strategy by attaching the target selection block and open the **Busy** treatment tab.
7. Add two **PlayApplication Busy** treatments with the custom `Param1` variable.
8. Configure the first treatment with the `{varStat1}` string and the second treatment with the `{VarStat2}` string.
9. Configure the other parameters for the target selection object, such as target, waiting time, and so on.
10. Save the strategy.

If the strategy runs and the target is not ready, the treatment request sent by URS for the first treatment will have the `Param1` parameter value of `StatExpectedWaitingTime` for the `VQ1_vit_sw2` switch (if it is not the first call), and for the second busy treatment, a value of 123.

Targets in Other Objects

Note: See the section “DN Target Type” on [page 589](#) for information on non-configurable target types. Also see function “`UseMediaType`” on [page 479](#).

In addition to the Routing objects, you can specify routing targets in some functions and/or treatment objects:

- Functions: `SelectDN` (see [page 608](#)) and `SuspendForDN` (see [page 614](#)) are alternative to the Routing Selection object. They provide the equivalent functionality to this object.
- `DeliverToIVR` function (see [page 593](#)) or IVR Treatment object (in `compatible` mode, see [page 409](#)). Both of them accept target where call should be temporary routed for applying IVR treatment.

- In addition there is set of functions that do not specify routing targets, but instead specify the properties that the routing targets must have. In this way, they affect target selection. They are UseDNTType (see [page 479](#)), UseMediaType (see [page 479](#)), and UseCustomDNTType (see [page 478](#)).

Segmentation Objects

Figure 188 on [page 374](#) shows the buttons for Segmentation objects that come into view when you click the Segmentation icon.

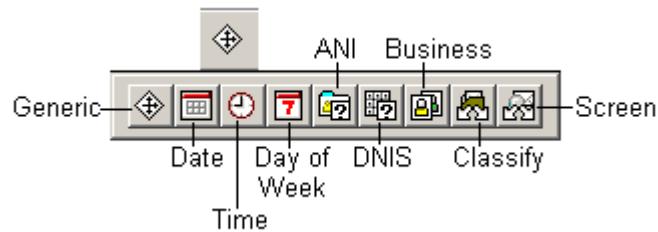


Figure 188: Icons for Segmentation Objects

Segmentation objects enable incoming interactions to take different paths. For example, you can segment based on the customer's revenue potential. Alternatively, you can segment interactions based on time, date, day of the week, customer phone number (ANI), number dialed (DNIS), or business rule. In releases 8.1 and later, the maximum number of entries in segmentation objects is increased to 100. This applies to any type of segmentation object.



ANI

The ANI Segmentation object separates incoming calls based on the Automatic Number Identification (ANI) number, which is the originating phone number from which the call was made. The `EventRouteRequest` message can contain ANI information:

```
EventRouteRequest*
AttributeDNIS '6661'
AttributeANI '5105551515'
AttributeThisDN '6661'
AttributeCollectedDigits '1'
```

Configure the value of the ANI for each segment in the ANI Segmentation Properties dialog box (see [Figure 189](#)).

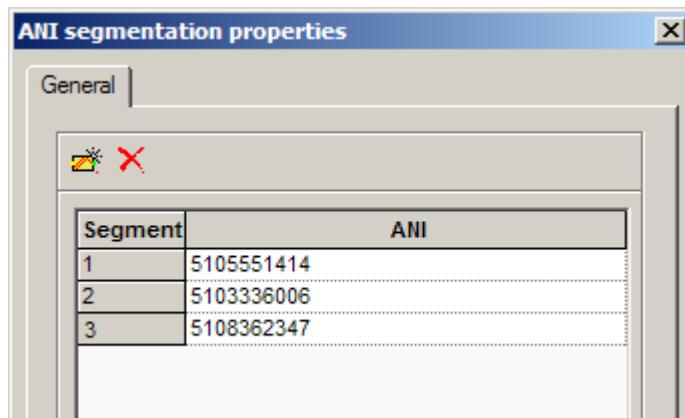


Figure 189: ANI Segmentation Properties Dialog Box



Business

The Business Segmentation object separates interactions based on existing business rules (see [page 71](#)). The example in [Figure 190](#) shows four business rules in the Business Segmentation Properties dialog box.

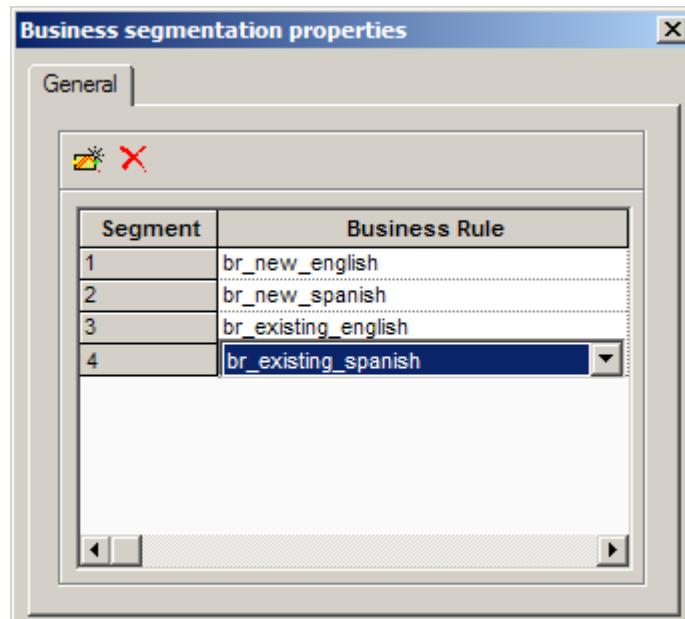
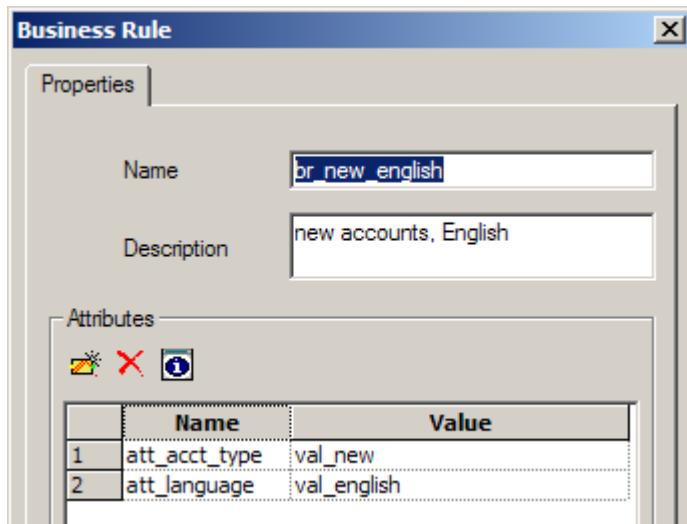


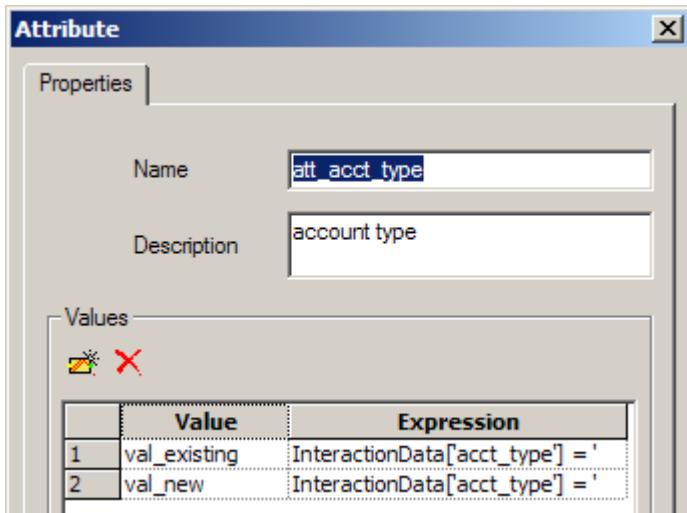
Figure 190: Business Segmentation Properties Dialog Box

Business rules are reusable objects (see [page 64](#)). In this example, the br_new_english business rule is defined using two attributes as shown in [Figure 191](#).

**Figure 191: Business Rule Dialog Box**

Attributes are reusable objects (see [page 64](#)), which you use to define business rules. The br_new_english business rule separates out interactions where the customer is new and their language preference is English.

Continuing with this example, each attribute in [Figure 191](#) (att_acct_type and att_language) is defined in IRD with two possible values, which are customer-entered digits (CEDs). [Figure 192](#) shows the values defined for att_acct_type.

**Figure 192: Attribute Dialog Box**

When you place a Business Segmentation object in a strategy, each rule creates its own output port. You can then connect each output from the Business Segmentation object to another object. [Figure 193](#) shows an example.

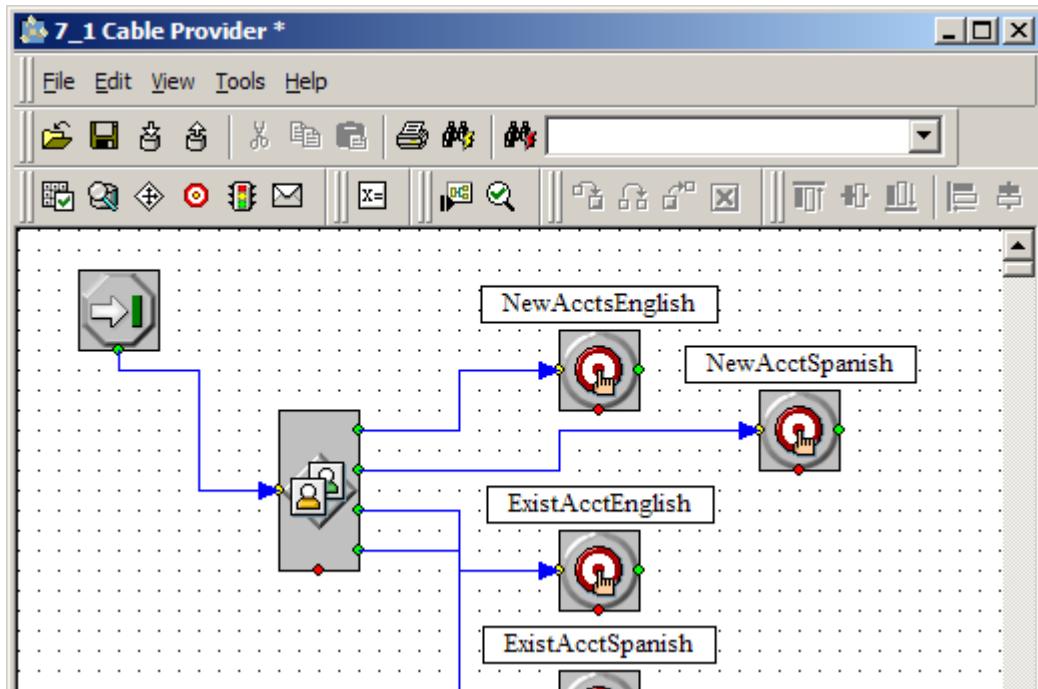


Figure 193: Business Segmentation Object Output Ports Connected



Classify

The Classify Segmentation object works with the Classify Multimedia object (see [page 198](#)) and the Attach Categories object (see [page 178](#)). The Classification Segmentation object segments incoming interactions to take different paths in a routing strategy based on classification categories.

This object provides a switching capability based on categories returned by Classification Server or assigned manually. When a strategy using this object is in the Routing Design window, the object block may have as many outlet ports as necessary for the strategy; for example as many ports as necessary for classification categories returned by the Classify object.

Classify General Tab

[Figure 194](#) shows the Classify Segmentation Properties dialog box when Classify by variable is selected.

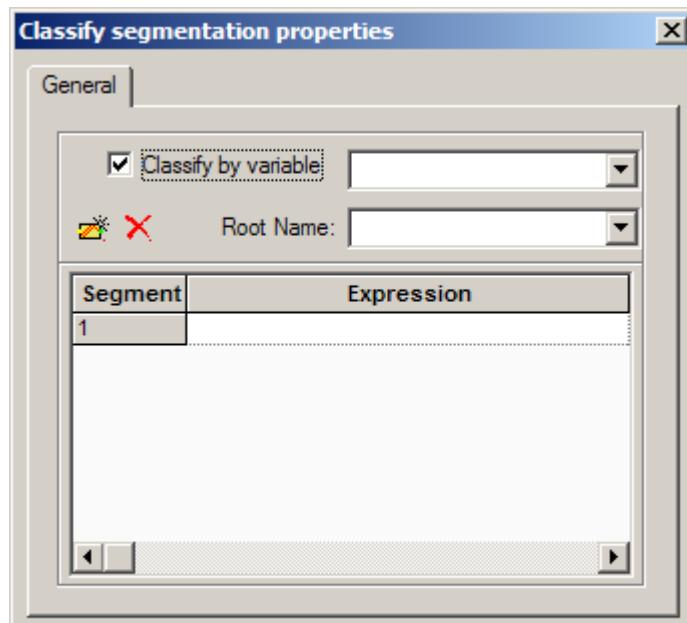


Figure 194: Classify Segmentation Object Properties Dialog Box

Example

The example below shows how the Classify and Classify Segmentation objects can work together.

- Assume the Classify object first initiates a Classify request using two categories as parameters: `cookery` and `fiction` (see [Figure 195](#)).

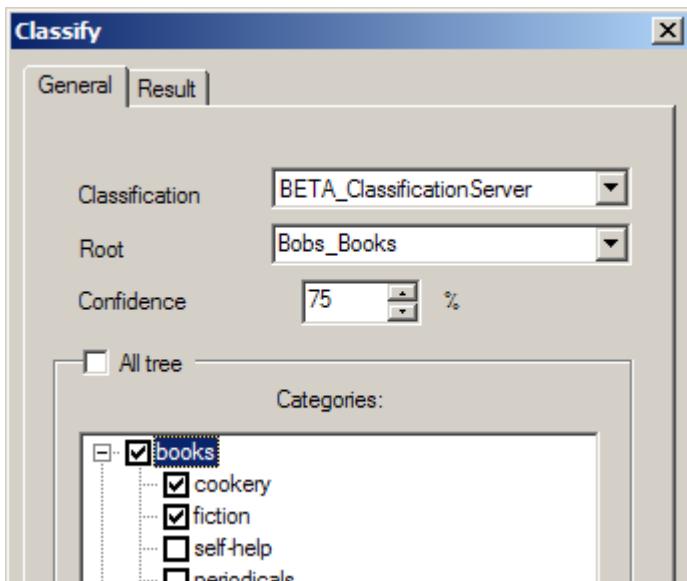


Figure 195: Example Classify Properties Dialog Box

URS attaches the classification results to the User Data with the key `CtgId`.

- The strategy also contains a Classify Segmentation object. It has two normal outgoing ports: one for `cookery` and another one for `fiction`. URS compares the identifier kept in the User Data with the key `CtgId` to the IDs of the two categories specified in the properties of the Classify Segmentation object (see [Figure 196](#)).

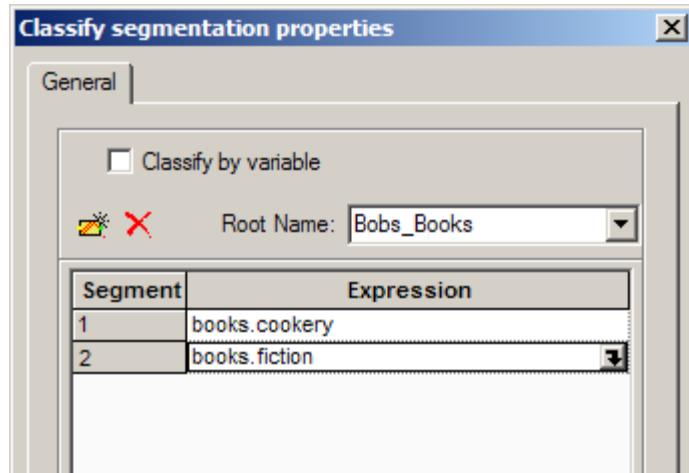


Figure 196: Example Classify Segmentation Properties Dialog Box

If `CtgId` doesn't exist or is empty, the interaction exits through the red port; otherwise the interaction exits through one of the green ports (see [Figure 197](#)).

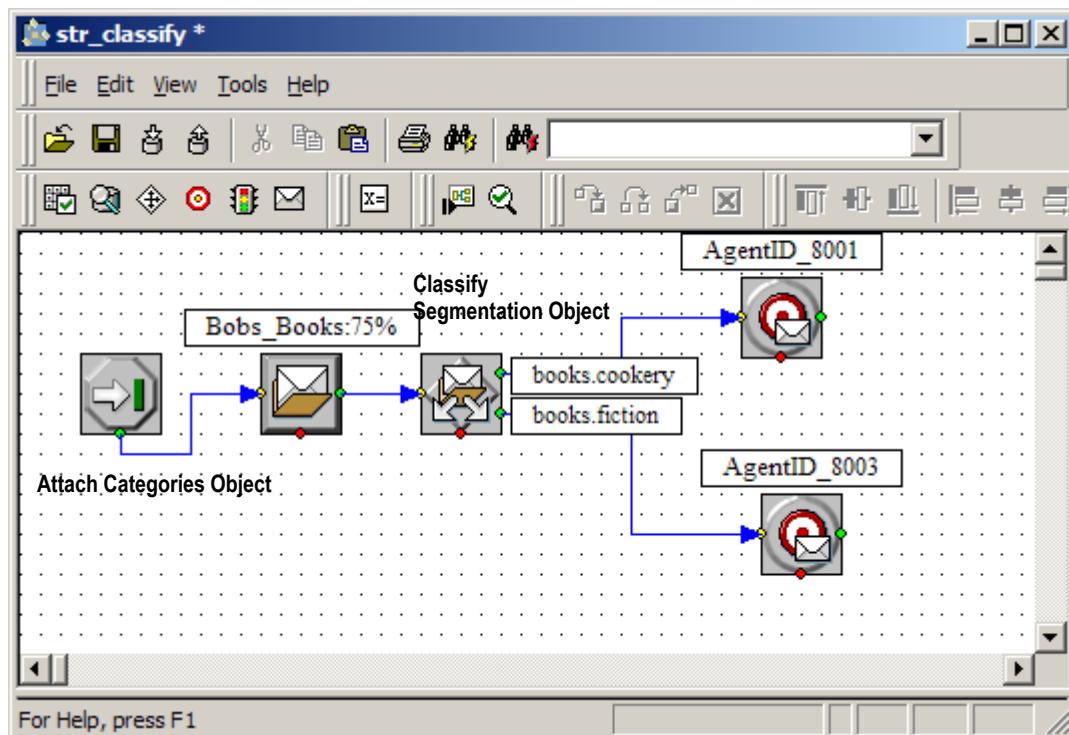


Figure 197: Classify and Classify Segmentation Objects



The Date Segmentation object enables you to separate interactions via from and to dates entered in the Date Segmentation Properties dialog box. When you select a time zone and enter one or more Date From and Up to date segments and click the OK button in the properties dialog box, the resulting Date Segmentation object contains the same number of output ports as segments (see [Figure 198](#)).

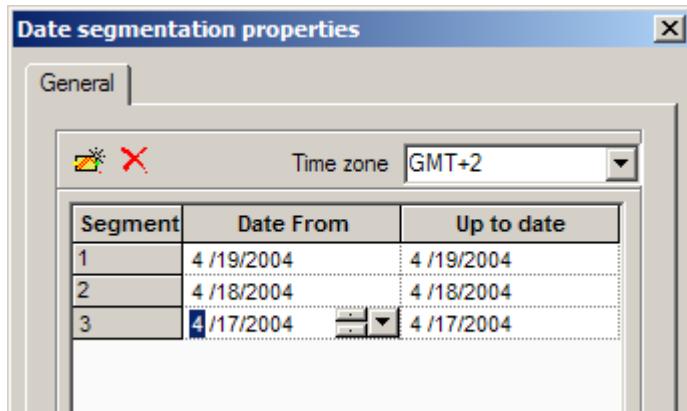


Figure 198: Date Segmentation Properties Dialog Box

Each interaction segment is then routed based on the date. The Check Integrity tool (see [page 34](#)) includes the time zone.

Note: The value of the Up To date parameter is excluded from the time interval. For example, if you segment interactions with a Date From of 10/21/07 and an Up To Date of 10/25/07, URS includes interactions arriving between 12:00:00 AM on 10/21/07 and 11:59:59 PM on 10/24/07.

This object only supports the following US date formats:

- M/d/yy
- M/d/yyyy
- MM/dd/yy
- MM/dd/yyyy
- yy/MM/dd
- dd-MMM-yy

Other formats prevent the strategy from compiling.





Day of Week

The Day of Week Segmentation object enables you to separate interactions based on the day of the week (Sunday through Saturday). As with the other Segmentation objects, you configure values in a properties dialog box (see Figure 199).

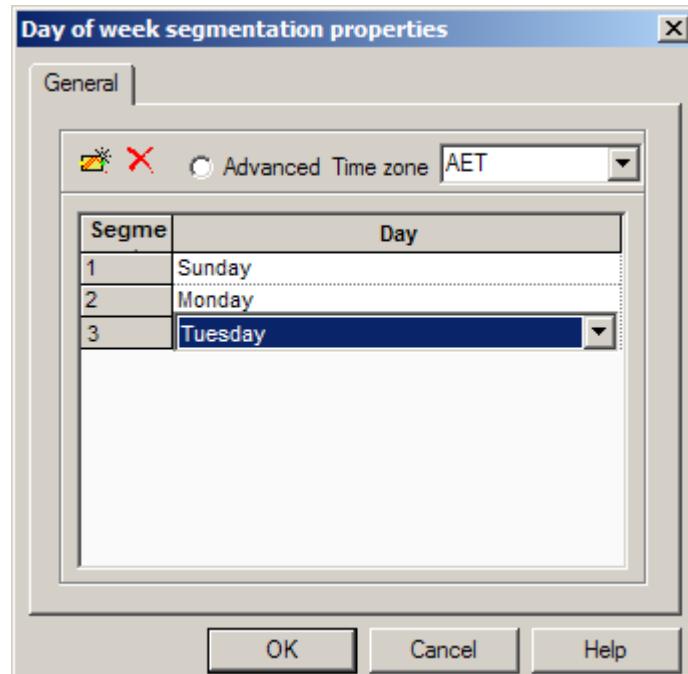


Figure 199: Day of Week Segmentation Properties Dialog Box

Within the General tab, you can create segments for a specific Day. For example, you can have interactions arriving on Saturday take a different path than interactions arriving on Sunday. You then connect each Segmentation object output port to an input port of the next object in the strategy.

By clicking the Advanced option in the dialog box, you can create segments for a range of days. For example, you can segment weekdays separately from weekends by creating two segments; one segment extends from Monday through Friday and another from Saturday through Sunday.

Multiple segments can be connected to the same object.

Note: The Check Integrity tool (see [page 34](#)) includes the time zone.



DNIS

The DNIS Segmentation object enables you to separate incoming calls based on the Dialed Number Identification System (DNIS) number. For example, it might be the 800 number that the caller dialed. The EventRouteRequest message can contain DNIS information:

```
EventRouteRequest*
AttributeDNIS'6661'
AttributeANI'5105551515'
AttributeThisDN'6661'
AttributeCollectedDigits'1'
```

Configure the value of the DNIS for each segment in the DNIS Segmentation Properties dialog box. [Figure 200](#) shows an example.

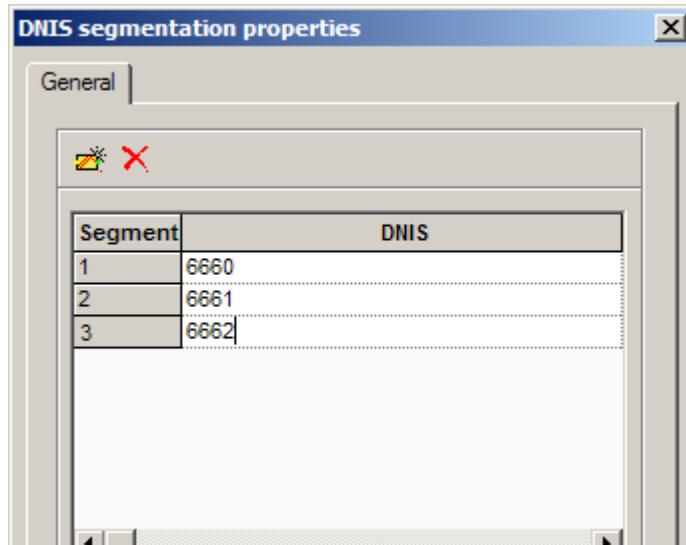
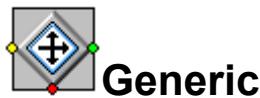


Figure 200: DNIS Segmentation Properties Dialog Box



The Generic Segmentation object enables you to segment interactions by configuring logical expressions in the Expression Builder, which opens when you click the Expression arrow in the Generic Segmentation Properties dialog box (see [Figure 201](#)).

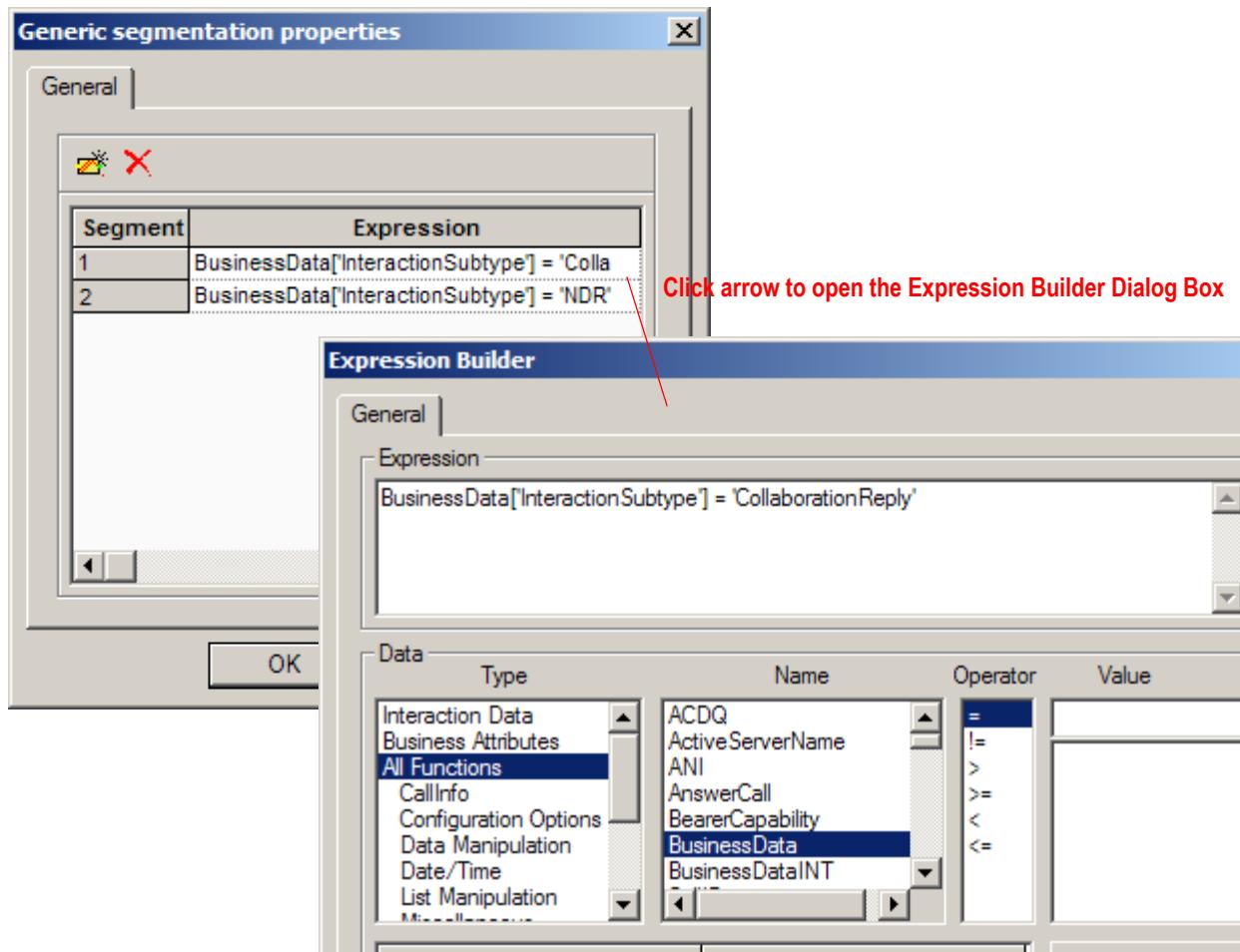
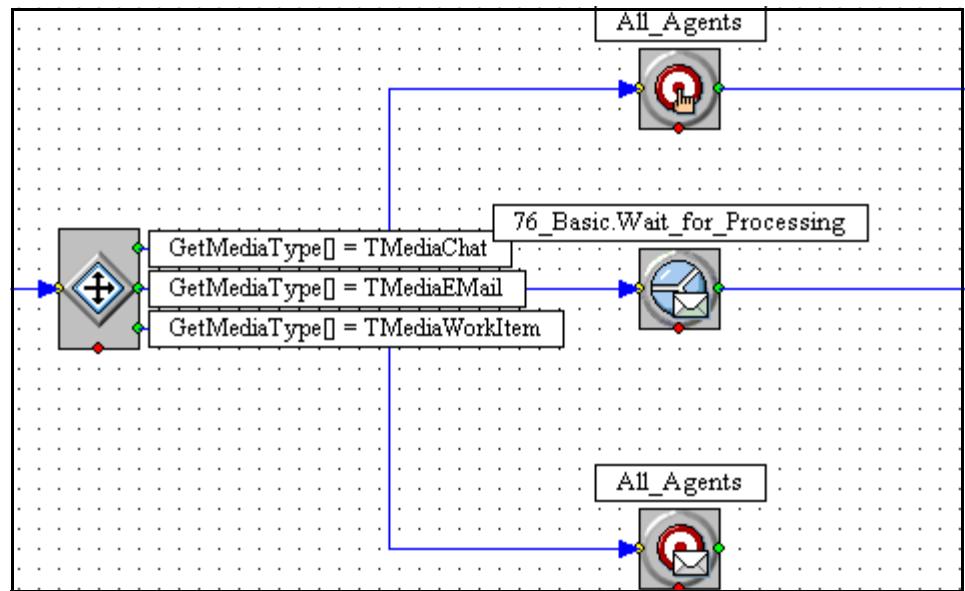


Figure 201: Generic Segmentation Properties Dialog Box

In Expression Builder, you select a data Type, Name, Operator, and Value (entered or selected). When you click Add, Verify, and OK in Expression Builder, the Generic Segmentation Properties dialog box contains the expression.

For example, you can configure three expressions and then click the OK button in the Generic Segmentation Properties dialog box (see [Figure 201](#)). The resulting Generic Segmentation object contains the same number of output ports as segments.

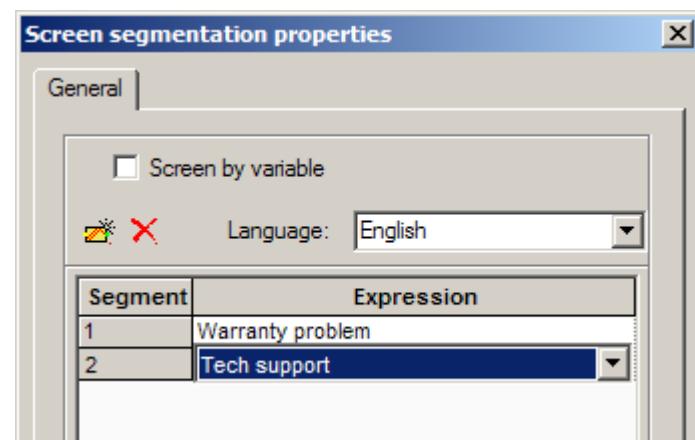
**Figure 202: Generic Segmentation Object in Strategy**

You then connect each output port to another object, such as a Routing object.

Note: Business rules and attributes can prevent having to repeatedly re-create logical expressions. See [page 71](#) for more information.



Use this object to process the results of the Screen (see [page 263](#)) or Multi-Screen object (see [page 241](#)). [Figure 203](#) shows the Screen Segmentation Properties dialog box when Screen by variable is selected.

**Figure 203: Screen Segmentation Properties Dialog Box**

The Screen Segmentation object segments incoming interactions to take different paths in the strategy based on Screening Rule matches.

Depending on how the Screen or MultiScreen object was previously used in the strategy, Screening Rule IDs may be written to interaction User Data (`AttributeUserData` as shown on [page 152](#)) or may be contained variables.

If you check `Screen by variable`, the variable must contain the Screening Rule ID to be checked.

Note: For additional information on Screening Rules and User Data, see the *eServices (Multimedia) 8.1 User's Guide*.



The Time Segmentation object enables you to separate interactions by selecting `From` and `Up until` times for each time segment in the Time Segmentation Properties dialog box (see [Figure 204](#)).

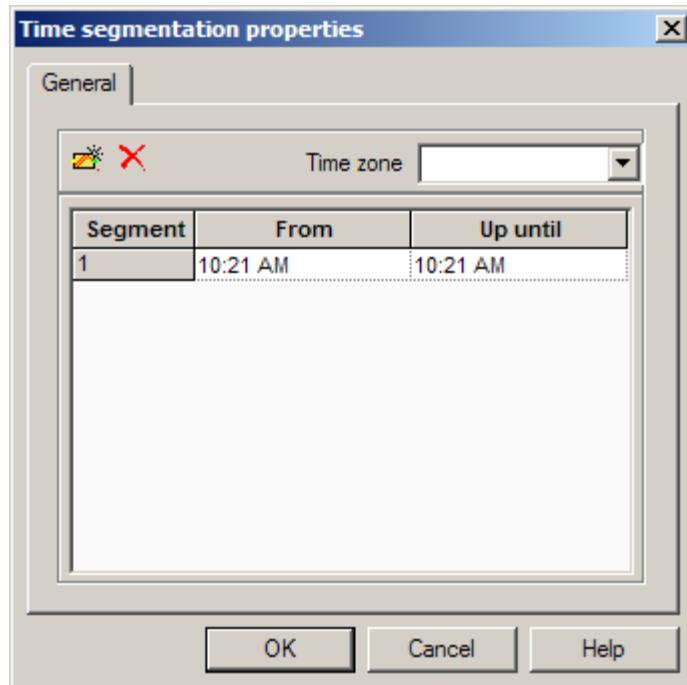


Figure 204: Time Segmentation Properties Dialog Box

When you select a time zone and enter one or more `From` and `Up until` rows and click the `OK` button in the properties dialog box, the resulting Time Segmentation object contains the same number of output ports as segments (see [Figure 204](#)). Each segment is then routed based on the time of day.

Note: In the Time Segmentation object, the value of the `Up until` parameter is excluded from the time interval. For example, if you segment interactions with a `From` time of 9:00:00 AM and an `Up until` time of 5:00:00 PM, URS includes interactions arriving between 9:00:00 AM and 4:59:59 PM.

The object only supports the following time format:

- hh:mm tt (where tt is AM or PM)

Note: The Check Integrity tool (see [page 34](#)) includes the time zone.

SMS Objects

[Figure 205](#) shows the buttons for SMS objects that come into view when you click the SMS icon.

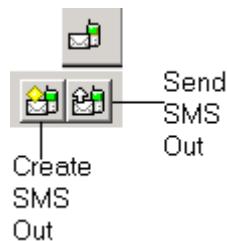


Figure 205: Icons for SMS Objects

SMS objects are strategy-building objects that create and send SMS messages by using ESP.



Create SMS Out

Use the Create SMS Out object to instruct SMS Server to create a new SMS interaction that will be handled by using ESP technology.

Note: Do not confuse this object with the Create SMS object on the Multimedia toolbar. The Multimedia Create SMS object creates a specially-configured e-mail-type message that is sent by the E-Mail Server. The Create SMS Out object uses ESP technology to create an SMS message that is sent by SMS Server.



Create SMS Out General Tab

Figure 206 on [page 387](#) shows an example completed General tab in the properties dialog box for the Create SMS Out object.

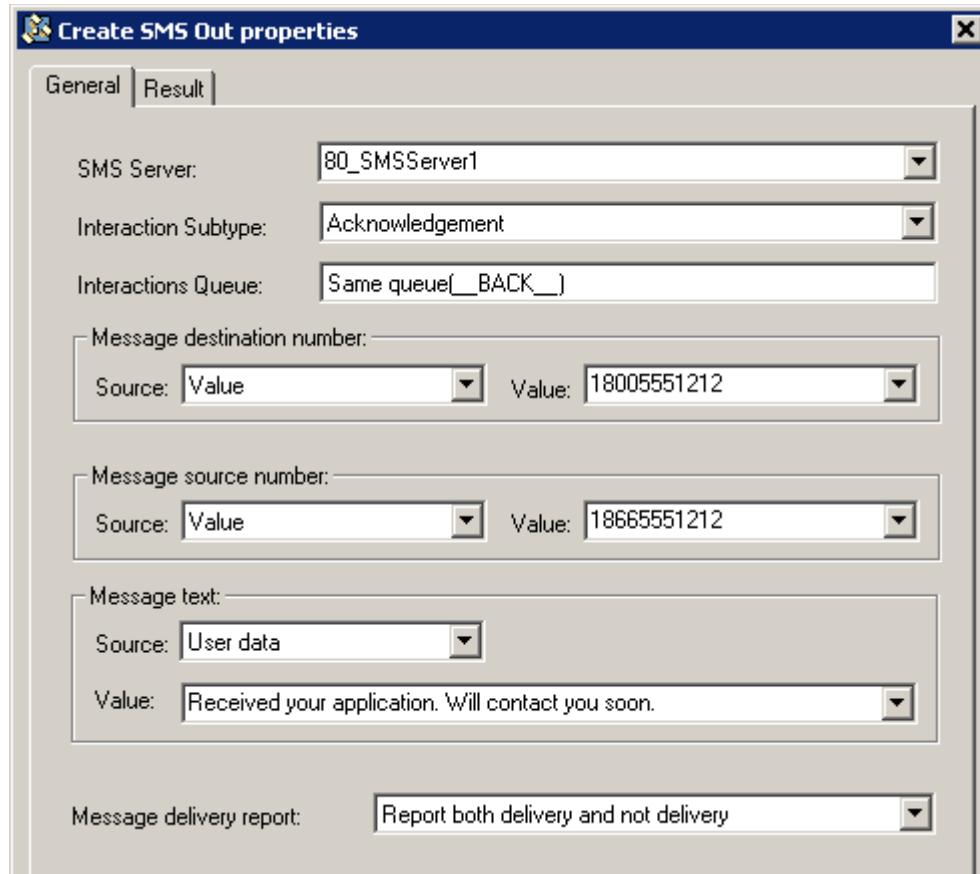


Figure 206: Create SMS Out Properties Dialog Box - General Tab

Use the information in Table 115 on [page 387](#) to complete the General tab of the properties dialog box.

Table 115: Create SMS Out General Tab

Parameter	Description
SMS Server	Enter the name of the SMS Server in the text box, or select the name from the drop-down list.
Interaction Subtype	Select the appropriate Interaction Subtype. The possible subtypes are Acknowledgement, Autoresponse, and Outbound New.
Interactions Queue	Click in the Interactions Queue text box to open a list of available queues. Select the queue from which to send the SMS message.

Table 115: Create SMS Out General Tab (Continued)

Parameter	Description
Message destination number	<p>Define the Message Destination Number. This is the mobile telephone number of the person to whom the message is to be sent:</p> <ol style="list-style-type: none"> 1. Enter or select the Source from the drop-down list. The possible sources are User Data and Value. 2. Enter (if you selected Value as the Source) or select (if you selected User Data as the Source) the Value for the Message Destination Number.
Message source number	<p>Define the Message Source Number. This is the mobile telephone number from which the SMS message should appear to come:</p> <ol style="list-style-type: none"> 1. Enter or select the Source from the drop-down list. The possible sources are User Data and Value. 2. Enter (if you selected Value as the Source) or select (if you selected User Data as the Source) the Value for the Message Source Number.
Message text	<p>Select or enter the Message Text. This is the content of the SMS message:</p> <ol style="list-style-type: none"> 1. Enter or select the Source from the drop-down list. The possible sources are User Data and Value. 2. Enter (if you selected Value as the Source) or select (if you selected User Data as the Source) the Value for the message text.
Message delivery report	<p>(Optional) Select the way you want to be notified of the SMS message's delivery outcome in the Message Delivery Report field.</p> <p>The choices are:</p> <ul style="list-style-type: none"> • No report required • Report both delivery and not delivery • Report not delivery only
Extra optional parameters	<p>The Extra Optional Parameters pane enables you to define additional message parameters or content, using previously configured key-value pairs:</p> <ol style="list-style-type: none"> 1. Click the Add Item icon to enter an extra parameter. 2. Click the X icon to delete any extra parameters you decide are unnecessary.

Create SMS Out Result Tab

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Use the information in Table 20 on [page 173](#) to complete the Result tab in the Create SMS Out Properties dialog box.

Returned Results for the Create SMS Out Object

The results returned will be either Event3rdServerResponse or Event3rdServerFault.

Any fault message returned uses the fault codes shown in [Table 116](#).

Table 116: Create SMS Out Object Fault Codes

FaultCode value	FaultString value	Description
900	<EVENT-SOURCE> exception <EVENT-DETAILED-DESCRIPTION>	This is a generic code/message, which may be returned on various abnormal situations during ESP request processing.
910	<EVENT-SOURCE> failed to find channel for outbound <DEST-NUMBER>, request ignored	SMS Server failed to find messaging channel (configuration section “channel-<name>”) for the specified destination number.
920	<EVENT-SOURCE> SMS message not delivered <REQUEST-REF-ID>	SMS Server failed to deliver an SMS message - no connection with SMSC, SMSC refused to accept the message, and so on.



Send SMS Out

Use this object to send an SMS message created by using the Create SMS Out object.

Send SMS Out General Tab

Figure 207 shows the Send SMS Out properties dialog box.

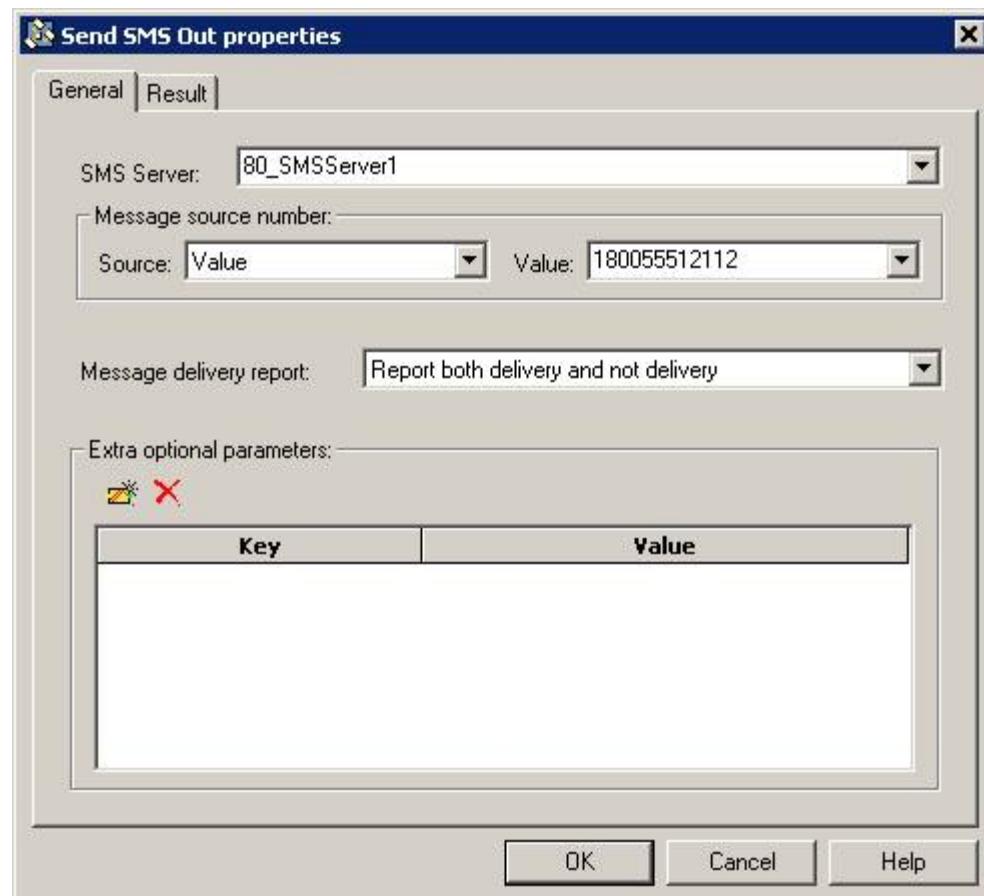


Figure 207: Send SMS Out Properties Dialog Box - General Tab

Use the information in Table 117 on [page 391](#) to complete the General tab of the Send SMS Out properties dialog box.

Table 117: Send SMS Out General Tab

Parameter	Description
SMS Server	Enter the name of the SMS Server in the text box, or select the name from the drop-down list.
Message source number	Define the Message Source Number. This is the mobile telephone number from which the SMS message should appear to come: <ol style="list-style-type: none"> 1. Enter or select the Source from the drop-down list. The possible sources are User Data and Value. 2. Enter (if you selected Value as the Source) or select (if you selected User Data as the Source) the Value for the Message Source Number.
Message delivery report	(Optional) Select the way you want to be notified of the SMS message's delivery outcome in the Message Delivery Report field. The choices are: <ul style="list-style-type: none"> • No report required • Report both delivery and not delivery • Report not delivery only
Extra optional parameters	The Extra Optional Parameters pane enables you to define additional message parameters or content, using previously configured key-value pairs: <ol style="list-style-type: none"> 1. Click the Add Item icon to enter an extra parameter. 2. Click the X icon to delete any extra parameters you decide are unnecessary.

Send SMS Out Result Tab

Note: IRD uses the same Result tab for the Acknowledgement, Autoresponse, Chat Transcript, Classify, Create Email Out, Create Interaction, Create Notification, Create SMS, Create SMS Out, Send SMS Out, External Service, Identify Contact, MultiScreen, Screen, and Submit New Interaction objects. Some options do not apply to all objects. Table 85 on [page 284](#) lists each object and indicates when an option does not apply.

Use the information in Table 20 on [page 173](#) to complete the Result tab in the Send SMS Out Properties dialog box.

Returned Results for the Send SMS Out Object

The results returned will be either Event3rdServerResponse or Event3rdServerFault.

Any fault message returned uses the fault codes shown in [Table 118](#).

Table 118: Send SMS Out Object Fault Codes

FaultCode value	FaultString value	Description
900	<EVENT-SOURCE> exception <EVENT-DETAILED-DESCRIPTION>	This is a generic code/message, which may be returned on various abnormal situations during ESP request processing.
910	<EVENT-SOURCE> failed to find channel for outbound <DEST-NUMBER>, request ignored	SMS Server failed to find messaging channel (configuration section “channel-<name>”) for the specified destination number.
920	<EVENT-SOURCE> SMS message not delivered <REQUEST-REF-ID>	SMS Server failed to deliver an SMS message - no connection with SMSC, SMSC refused to accept the message, and so on.

Workflow and Resource Management Objects

Figure 208 on [page 392](#) shows the buttons for Workflow and Resource Management objects that come into view when you click the Workflow and Resource Management icon.

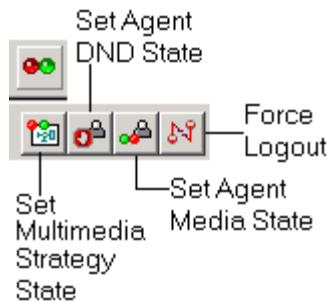


Figure 208: Icons for Workflow and Resource Management Objects

Workflow and Resource Management objects are strategy-building objects that enable you to control certain settings for strategies, Places, and Agents.



Force Logout

Use the Force Logout object to log out a Place or an Agent when an interaction is directly routed to the Agent DN rather than through the use of targets.

Force Logout Properties Dialog Box

Figure 209 shows the Force Logout properties dialog box.

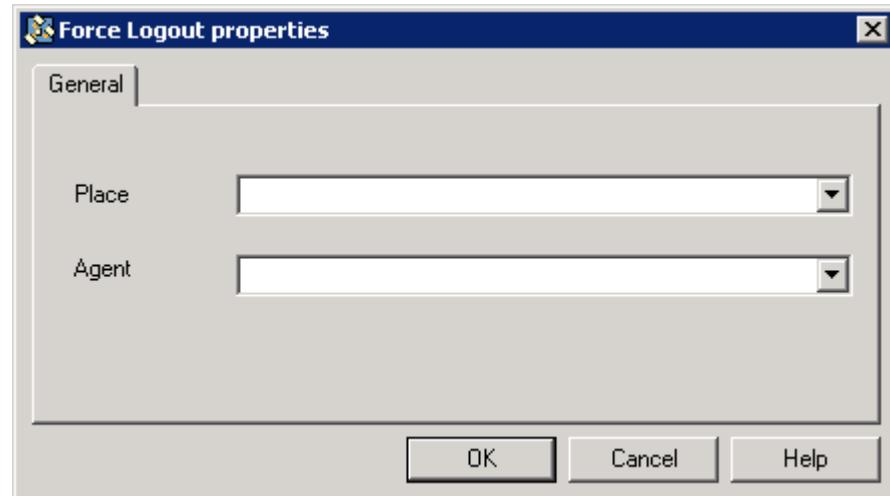


Figure 209: Force Logout Properties Dialog Box

Use the information in Table 119 on [page 393](#) to complete the General tab of the Force Logout properties dialog box. You enter the Place or the Agent that is to be logged out.

Table 119: Force Logout Properties Dialog Box

Parameter	Description
Place	To set the Place that is to be logged out, enter the Place name in the Place text box, or click the drop-down arrow to select the Place from a list. If you click the drop-down arrow, you can choose to select from a list of Places or variables.
Agent	To set the Agent who is to be logged out, enter the Agent ID in the Agent text box, or click the drop-down arrow to select the Agent ID from a list. If you click the drop-down arrow, you can choose to select from a list of Agent IDs or variables.

Returned Results for the Force Logout Object

The results returned will be either a response message (`Event3rdServerResponse`) or a fault message (`Event3rdServerFault`). Any fault message returned uses the fault codes in [Table 120](#).

Table 120: Force Logout Object Fault Codes

FaultCode value	FaultString value	Description
106	Invalid Request	Invalid method.
201	Missing parameter name	One parameter is missing
206	Unknown tenant	The tenant can not be found
213	Invalid type of <parameter name>	The Type is not set to either Place or Variable
218	Unknown value of parameter 'State'	The State parameter has an invalid value
229	Neither 'Place' nor 'Agent' is specified in the request	A request was attempted without specifying a value for Place or Agent
230	This agent is not logged in	The Agent ID specified for forced logout is for an agent who is currently not logged in
501	Place not found	The Place specified in the request could not be found or is invalid
502	Place disabled	The Place specified in the request is disabled
503	Place busy	The Place specified in the request is currently busy and can not be logged out
505	Agent disabled	The Agent ID specified in the request is for an agent that is currently not enabled
507	Place not occupied	The Place specified in the request is not occupied

Table 120: Force Logout Object Fault Codes (Continued)

FaultCode value	FaultString value	Description
508	Place not occupied by specified agent	The Place specified in the request is not occupied by the Agent ID also specified in the request.
509	Place not occupied by anonymous	The Place specified in the request is not occupied by an anonymous agent



Set Agent DND State

Use the Set Agent DND object to turn the Agent Do-Not-Disturb (DND) value On or Off when an interaction is directly routed to the Agent DN rather than through the use of targets.

Set Agent DND State Properties Dialog Box

Figure 210 shows the Set Agent DND State properties dialog box.

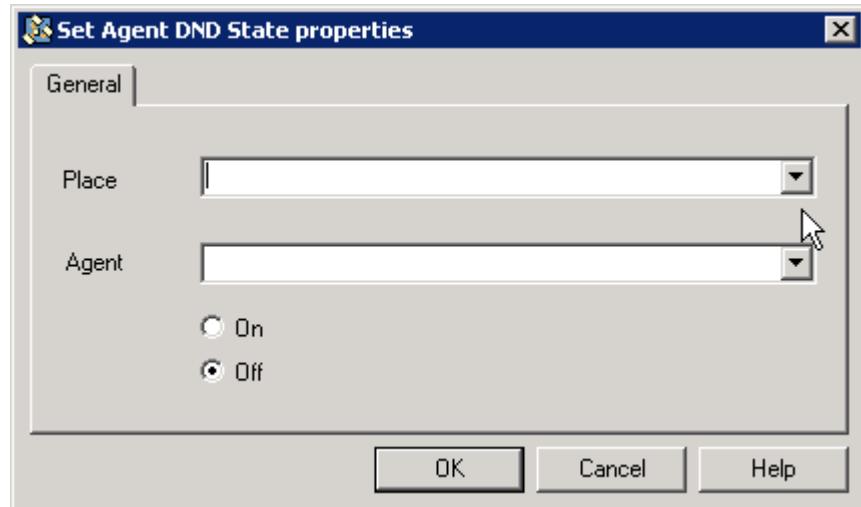


Figure 210: Set Agent DND State Properties Dialog Box

Use the information in [Table 121](#) to complete the General tab of the Set Agent DND State properties dialog box. You enter the Place or the Agent whose DND state is to be changed.

Table 121: Set Agent DND State Properties Dialog Box

Parameter	Description
Place	To set the Place in which the DND state is to be changed, enter the Place name in the Place text box, or click the drop-down arrow to select the Place from a list. If you click the drop-down arrow, you can choose to select from a list of Places or variables.
Agent	To set the Agent whose DND state is to be changed, enter the Agent ID in the Agent text box, or click the drop-down arrow to select the Agent ID from a list. If you click the drop-down arrow, you can choose to select from a list of Agent IDs or variables.
On/Off	Click the radio button that corresponds to the DND state (On or Off) that you want to set for this Place or Agent.

Returned Results for the Set Agent DND State Object

The results returned will be either a response message (`Event3rdServerResponse`) or a fault message (`Event3rdServerFault`). Any fault message returned uses the fault codes in [Table 122](#).

Table 122: Set Agent DND State Object Fault Codes

FaultCode value	FaultString value	Description
106	Invalid Request	Invalid method.
201	Missing parameter name	One parameter is missing
206	Unknown tenant	The tenant can not be found
213	Invalid type of <parameter name>	The Type is not set to either Place or Variable
218	Unknown value of parameter 'State'	The State parameter has an invalid value
229	Neither 'Place' nor 'Agent' is specified in the request	A request was attempted without specifying a value for Place or Agent

Table 122: Set Agent DND State Object Fault Codes (Continued)

FaultCode value	FaultString value	Description
230	This agent is not logged in	The Agent ID specified for DND state change is for an agent who is currently not logged in
501	Place not found	The Place specified in the request could not be found or is invalid
502	Place disabled	The Place specified in the request is disabled
503	Place busy	The Place specified in the request is currently busy and can not be logged out
505	Agent disabled	The Agent ID specified in the request is for an agent that is currently not enabled
507	Place not occupied	The Place specified in the request is not occupied
508	Place not occupied by specified agent	The Place specified in the request is not occupied by the Agent ID also specified in the request.
509	Place not occupied by anonymous	The Place specified in the request is not occupied by an anonymous agent
516	DND already on	A request was made to set DND to <code>on</code> , but DND is already on.
517	DND already off	A request was made to set DND to <code>off</code> , but DND is already off.



Set Agent Media State

Use the Set Agent Media State object to change the Agent state from Not Ready to Ready—or vice versa—for a specified Media Type when an interaction is directly routed to the Agent DN rather than through the use of targets.

Set Agent Media State Properties Dialog Box

Figure 211 on [page 398](#) shows the Set Agent Media State properties dialog box.

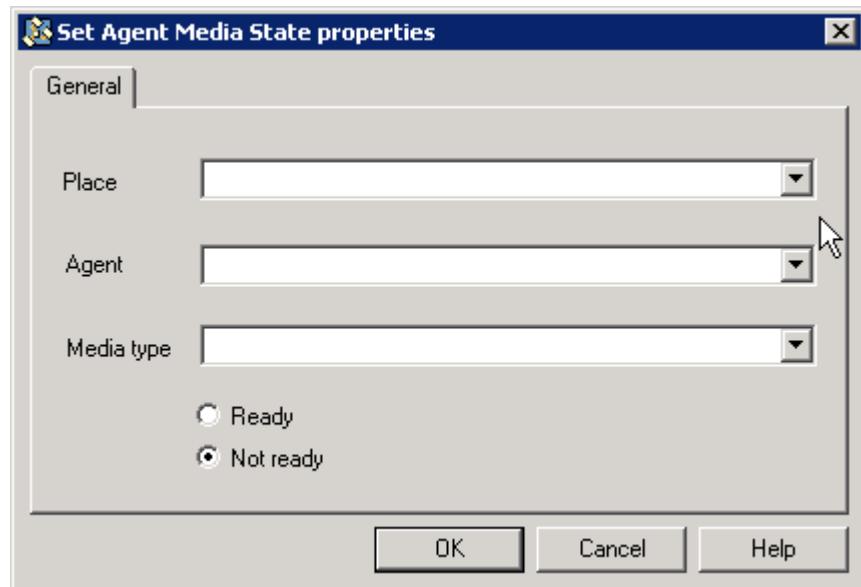


Figure 211: Set Agent Media State Properties Dialog Box

Use the information in [Table 123](#) to complete the General tab of the Set Agent Media State properties dialog box. You enter the Place or the Agent whose media state is to be changed.

Table 123: Set Agent Media State Properties Dialog Box

Parameter	Description
Place	To set the Place in which the media state is to be changed, enter the Place name in the Place text box, or click the drop-down arrow to select the Place from a list. If you click the drop-down arrow, you can choose to select from a list of Places or variables.
Agent	To set the Agent whose media state is to be changed, enter the Agent ID in the Agent text box, or click the drop-down arrow to select the Agent ID from a list. If you click the drop-down arrow, you can choose to select from a list of Agent IDs or variables.



Table 123: Set Agent Media State Properties Dialog Box (Continued)

Parameter	Description
Media type	Enter the media type whose state is to be changed, or click the drop-down arrow to select the media type from a list. If you click the drop-down arrow, you can choose to select from a list of media types or of variables.
Ready/Not ready	Click the radio button that corresponds to the state (Ready or Not Ready) that you want to set for this Place or Agent.

Returned Results for the Set Agent Media State Object

The results returned will be either a response message (`Event3rdServerResponse`) or a fault message (`Event3rdServerFault`). Any fault message returned uses the fault codes in [Table 124](#).

Table 124: Set Agent Media State Object Fault Codes

FaultCode value	FaultString value	Description
106	Invalid Request	Invalid method.
201	Missing parameter name	One parameter is missing
206	Unknown tenant	The tenant can not be found
213	Invalid type of <parameter name>	The Type is not set to either Place or Variable
218	Unknown value of parameter 'State'	The State parameter has an invalid value
229	Neither 'Place' nor 'Agent' is specified in the request	A request was attempted without specifying a value for Place or Agent
230	This agent is not logged in	The Agent ID specified for media state change is for an agent who is currently not logged in
501	Place not found	The Place specified in the request could not be found or is invalid
502	Place disabled	The Place specified in the request is disabled

Table 124: Set Agent Media State Object Fault Codes (Continued)

FaultCode value	FaultString value	Description
503	Place busy	The Place specified in the request is currently busy and can not be logged out
505	Agent disabled	The Agent ID specified in the request is for an agent that is currently not enabled
507	Place not occupied	The Place specified in the request is not occupied
508	Place not occupied by specified agent	The Place specified in the request is not occupied by the Agent ID also specified in the request.
509	Place not occupied by anonymous	The Place specified in the request is not occupied by an anonymous agent
514	Agent does not have the media	The media for which the state change was requested is not a valid media for the specified Agent ID
518	Agent is already ready for media	A request was made to set media state to Ready, but media state is already set to Ready.
519	Agent is already not ready for media	A request was made to set media state to Not Ready, but media state is already set to Not Ready.



Set Multimedia Strategy State

Use the Set Multimedia Strategy State object as a method for managing workflows and changing the state of multimedia strategies. Strategies can be activated or deactivated by an interaction that has specific properties. This object cannot be used to change the state of voice strategies.



When activating a strategy using the Set Multimedia Strategy State object, the interaction server finds all switches for the tenant, then either creates Virtual Routing Points (VRPs) in all the switches, or updates them if they already exist.

When deactivating a strategy using the Set Multimedia Strategy State object, the interaction server finds all switches for the tenant and deletes from their DN lists all VRPs matching the StrategyName.

When unloading a strategy using the Set Multimedia Strategy State object, the interaction server finds VRPs matching the StrategyName in all switches for the tenant and modifies the Annex in these VRPs accordingly.

Set Multimedia Strategy State Properties Dialog Box

Figure 212 shows the Set Multimedia Strategy State properties dialog box.

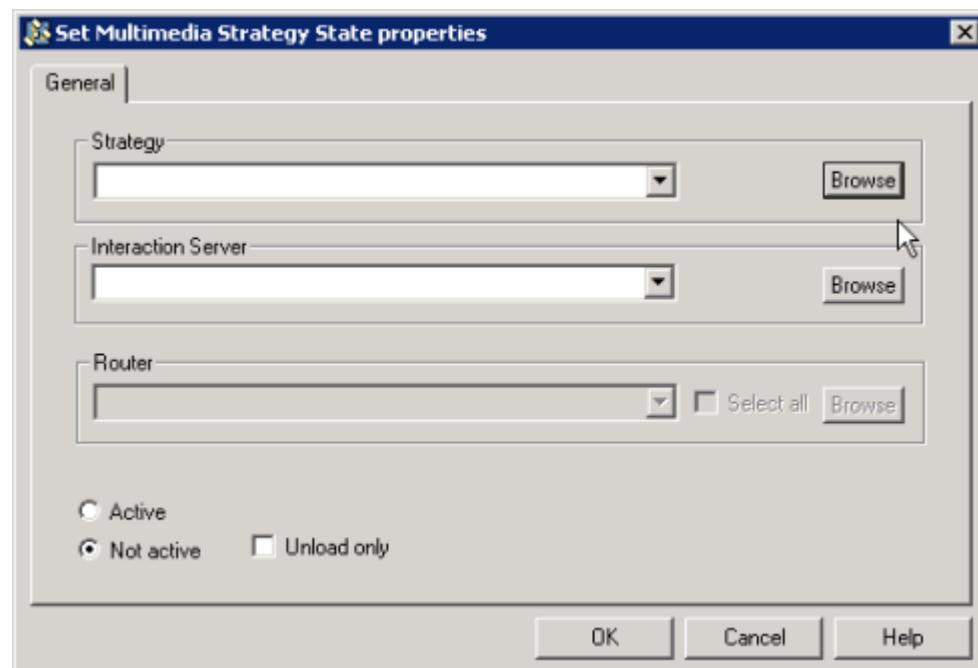


Figure 212: Set Multimedia Strategy State Properties Dialog Box

Use the information in [Table 125](#) to complete the General tab of the Set Multimedia Strategy State properties dialog box.

Table 125: Set Multimedia Strategy State Properties Dialog Box

Parameter	Description
Strategy	Select a strategy from the drop-down list, or click Browse to select from the complete list of strategies for the current Tenant from the Select Strategy dialog box. Click a column header to re-sort the list or enter text in a filter text box to find matching text in a strategy name, description, or other column.
Interaction Server	Select the Interaction Server from the drop-down list, or click Browse to select from the complete list of available Interaction Servers from the Select Interaction Server dialog box.
Router	If available, select the Router from the drop-down list, or click Browse to select from the complete list of available Routers from the Select Router dialog box. Use the Select All check box to select all Routers.
Active/Not active	Select either the Active or the Not Active radio button. The Active button activates the strategy; the Not Active button deactivates it.
Unload only	Select the Unload Only check box to deactivate the strategy without removing the routing point to which it was loaded.

Returned Results for the Set Multimedia Strategy State Object

The results returned will be either a response message (`Event3rdServerResponse`) or a fault message (`Event3rdServerFault`). Any fault message returned uses the fault codes in [Table 126](#).

Table 126: Set Multimedia Strategy State Object Fault Codes

FaultCode value	FaultString value	Description
106	Invalid Request	Invalid method.
201	Missing parameter name	One parameter is missing
206	Unknown tenant	The tenant can not be found
213	Invalid type of <parameter name>	The Type is not set to either Place or Variable

Table 126: Set Multimedia Strategy State Object Fault Codes (Continued)

FaultCode value	FaultString value	Description
218	Unknown value of parameter 'State'	The State parameter has an invalid value
219	Unknown (tenant, strategy name) specified in ESP request	The strategy name or Interaction Server name specified in the request can not be found or is invalid
220	Unknown value of parameter 'UnloadOnly'	An error has occurred in which the value of the Unload Only parameter can not be determined
221	The specified T-Server or Interaction Server has no switches to (un)load the strategy	The T-Server or Interaction Server does not have an identifiable switch to load or unload the requested strategy
222	This tenant has no routers connected to the specified T-Server or Interaction Server	The tenant does not have a router connected to the T-Server or Interaction Server.
223	There are no VRPs found that match the strategy name	The specified strategy name can not be found on any available Virtual Routing Point
224	Failed to delete VRP in ConfigServer	The Virtual Routing Point could not be deleted from Configuration Server
225	Failed to add new VRP in ConfigServer	The Virtual Routing Point could not be added to Configuration Server
226	Failed to update a DN in ConfigServer	The DN could not be updated in Configuration Server
227	The strategy is already fully active on all specified routers.	A request was made to activate a strategy, but the strategy is already fully active

Table 126: Set Multimedia Strategy State Object Fault Codes (Continued)

FaultCode value	FaultString value	Description
228	The strategy is not loaded on this DN	A request was made to change strategy state on a DN that does not have the strategy loaded
234	This tenant has no switches connected to the specified Interaction Server	
236	No routers connected to the Interaction Server were found in tenant	
237	No router of the specified name and connected to the Interaction Server was found in tenant	
239	No DN of the specified name found among DNs of types: RoutingPoint, VRP, RoutingQueue, ServiceNumber, VoiceTreatmentPort	
240	Some other strategy is already loaded on this DN	
241	The 'TServerName' parameter must be specified if RouterName is provided	

Voice Treatment Objects

Figure 213 shows the buttons for Voice Treatment objects that come into view when you click the Voice Treatments icon.

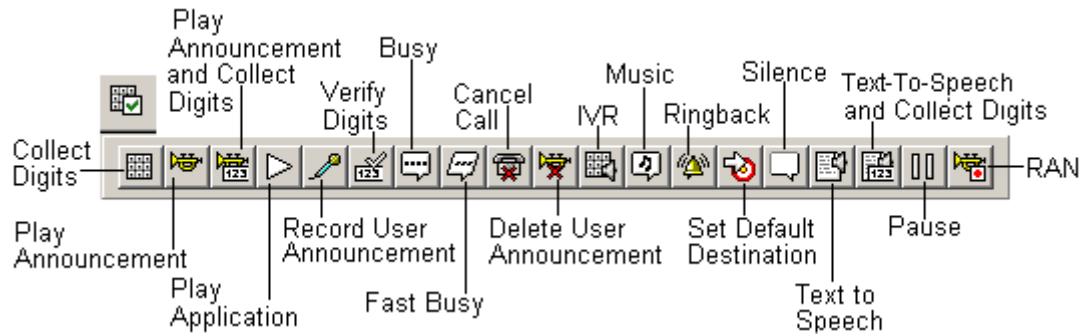


Figure 213: Voice Treatment Object Toolbar

Note: Some switches do not support specific treatment calls. For a list of supported treatment functions, refer to the Supported Functionality in T-Server chapter in your Genesys T-Server documentation. For example, refer to the Supported Functionality in T-Server for Alcatel A4400 chapter in the *T-Server for Alcatel A4400 Reference Manual*.

Mandatory Versus Busy

Mandatory treatments are those to which an interaction will be subject whether there are any available targets or not. In IRD, all Mandatory treatments are represented as objects. Mandatory IVR (Interactive Voice Response) treatments can often be used in determining the appropriate target for the interaction.

Busy treatments, discussed on [Page 357](#), are applied when all the targets selected are busy and the interaction is waiting for an available target.

Mandatory treatments are played through only once before the interaction continues on through the strategy. Busy treatments, which are configured as a property of many of the Routing objects, can play repeatedly while an interaction is waiting for an available target and include a timeout.

Warning! When specifying busy treatment parameters, do not include pipes “|” and colons “:” in values.

Updating Wait Time Information for an IVR

The SetStatUpdate function enables you to periodically update statistical data within an interaction without leaving the target selection object. The SetStatUpdate function supports (but is not limited to) use of the following statistics:

- PositionInQueue
- CallsWaiting

- InVQWaitTime (for virtual queues only).

See [page 575](#) for more information.

Beginning a Busy Treatment

The `compat_treatments` option in the URS Application object affects the way URS requests T-Server to begin a busy treatment. See [page 636](#) for more information.

Voice Treatment Parameters

Parameters for each object enables you to specify how each treatment functions. You can enter STRING or INTEGER values (depending on the parameter) and variables.

Note: You cannot use variables for Busy treatments in routing rules because variables are specific to a strategy and routing rules are not.

Variables must be previously defined in the Variable list. When using a variable, select the variable from the drop-down list or type the variable name manually. If entered manually, IRD searches for the variable from the Variable list and selects the one with the same name. If the variable is not found, IRD interprets the variable as a string.

Switch API Settings

IRD provides Voice Treatment objects for all systems. However, some switch APIs (application programming interfaces) do not support all available voice treatments. If you set the `Compatible` mode (set the `compat_treatments` option to `true`, as described on [page 636](#)), the `TGiveTreatment` API is used; otherwise the `TApplyTreatment` API is used. See *Genesys Events and Models Reference Manual* for more information.

For the `TGiveTreatment` switch API, select `Compatible` mode in the `Treatment Properties` dialog box. The following voice treatments can be used in `Compatible` mode:

- IVR
- Music
- Ringback
- Silence

Note: The IVR voice treatment is different from the other treatments because it routes the interaction to an IVR. All other treatments work at the switch or the intelligent peripheral.

Voice Treatment Error Messages

The following error messages are possible for Voice Treatment objects:

- **0001** Unknown error
- **0003** Treatment in Progress
- **0004** Call is on hold
- **0005** Call is gone
- **0008** Routing done
- **0013** Remote error—error/unknown message from DB Server

The Voice Treatment Objects

This section lists the voice treatment objects (see [Figure 213](#)) in alphabetical order and describes the parameters that you configure in the Properties dialog box for each object.



This treatment is used to connect the interaction to a source of busy tone.

Note: On a DMS-100 switch, this treatment terminates the interaction—the caller hears a busy signal, but the interaction can no longer be transferred.

Parameter:

DURATION The time, in seconds, that the treatment is applied.



This treatment is used to disconnect an interaction in progress. This treatment ends the strategy and deletes the interaction from URS memory.

Parameters:

None



This treatment is used to collect digits from the caller.

Parameters:**MAX_DIGITS**

Maximum number of digits to be collected (up to 31).

Note: Maximum digits may be equal to 0. In this case, no time is spent waiting for caller input digits, and a response is returned indicating 0 digits collected.

ABORT_DIGITS

This sequence of up to two keys aborts the digit-collection operation. If this sequence appears, the intelligent peripheral considers this to be a failed digit-collection attempt.

IGNORE_DIGITS

This sequence of up to two keys is treated as though the keys have not been pressed.

BACKSPACE_DIGITS

This sequence of up to two keys causes the previous keystroke to be discarded.

TERM_DIGITS

This sequence of up to two keys causes all the digits, not including the TERM_DIGITS, to be returned to the service logic as collected digits.

RESET_DIGITS

This sequence of up to two keys causes all the previous keystrokes to be discarded. The digit collection resumes.

CLEAR_FLAG

Indicates whether any information that has been input should be cleared before digit collection starts. The type of the value is Boolean (0/1). Not supported in GR-1129-CORE protocol implementation.

START_TIMEOUT

The number of seconds the IP should wait for the caller to begin DTMF input.

DIGIT_TIMEOUT

The number of seconds the IP should wait between DTMF digits.

TOTAL_TIMEOUT

The total number of seconds the IP should wait for the caller to provide the requested DTMF input.



Delete User Announcement

This treatment is used to remove a user-specific announcement from an IP.

Parameters:

USER_ID	User ID string specifying the user for an announcement to be deleted.
USER_ANN_ID	User Announcement ID (integer) as returned in the EventTreatmentEnd event after a successful RecordUserAnnouncement request.



Fast Busy

Provides a different busy signal from the one in a Busy treatment. This treatment is supported on a limited number of switches.

Note: On a DMS-100 switch, this treatment terminates the interaction. The caller hears a fast busy signal, but the interaction can no longer be transferred.

Parameters:

DURATION	The time, in seconds, that the treatment is applied.
----------	--



IVR

This treatment is used to connect the interaction to the IVR. The IVR treatment is specified in a format similar to that of a target because URS transfers the interaction in the same way as it would when routing to a target. IVR produces a set of DNs from the Source (corresponding to a target ID), location, and type, and from Stat Server information. If any of these DNs are available, it sends the interaction to one of them. URS does not verify that the DN to which it sends the interaction is an IVR. For example, it may be useful to send the interaction to a queue, which distributes it to the actual IVR.

When choosing a target for an IVR, select the target type and specify all parameters necessary for that type. See “Targets in Other Objects” on [page 373](#) for more information on target types.

In an IVR treatment, do not set type to Agent (.A) or Group of Agents (.GA).

The IVR itself is often configured in Configuration Layer as a DN of type Voice Treatment Port, where VTO (Voice Treatment Option) scripts can run.

The most frequent configuration for using IVRs in a strategy involves creating a Place in Configuration Layer and creating a shortcut from the place to the

IVR DN. Then IVRs can be used as treatments by specifying the place or group of places that corresponds to an IVR.

Another way to specify the IVR is to check Compatible Mode in the dialog box and then manually enter the DN as a source and the name of the switch as the location (<DN>@<name of switch>) in the Target field.

Note: When the URS configuration option `compat_treatments` is set to `false` (needed for Network T-Servers), you can enter a duration for the IVR treatment, but should not configure any of the remaining parameters. T-Server presumes that, in this case, the default routing object is the IVR in question. URS sends all treatment requests using the `TApplyTreatment` function. When Compatible Mode is selected, URS is forced to send treatment requests in a mode compatible with IR 5.0.1.

All voice treatments except IVR work at the switch or the intelligent peripheral. The IVR treatment is different from the other treatments because URS routes the interaction to an IVR, just as it does for a target. Therefore, the format used to specify the IVR treatment is similar to that for a target.

Parameters:

LABEL	Routing target address if connection to T-Server is lost.
DNIS	Value is used instead of existing DNIS if connection to T-Server is lost.

Parameters for Compatible Mode:

SCRIPT	Script name or variable for the script name
TARGET	The target type (Agent, Agent Group, Destination Label, Routing Point, Place, Place Group, Queue, and Variable) for this treatment, target name, and location for the Stat Server.
DURATION	The time, in seconds, that the treatment is applied.

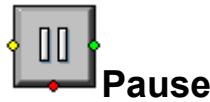


Music

This treatment is used to connect the interaction to a music source.

Parameters:

MUSIC_DN	Directory number of the music source.
DURATION	Music duration in seconds. This parameter is optional.



For this treatment, specify its duration only in seconds. It is the only treatment that does not send a command to T-Server. If a treatment other than an IVR precedes it, that treatment continues for the duration of the Pause treatment. In the case of a preceding IVR treatment, the Pause treatment starts only after the interaction has been returned to the routing point. Then the caller hears nothing for the specified time interval.

Parameters:

DURATION The time, in seconds, that the treatment is applied.



This treatment is used to play an announcement block to the calling party. The entire announcement block can consist of a series of Announcement Elements pieced together. Each Announcement Element can be described as interruptible or non-interruptible.

Parameters:

PROMPT Contains 10 possible sublists, numbered 1 to 10. Each sublist contains entries describing an announcement element, which are as follows:

- Interruptible (check box) indicates whether the caller can interrupt the announcement. IRD instructs URS to send a message to T-Server indicating that the announcement is interruptible (1) or not (0).

Specify one of the following options:

- IDinteger ID of a message to play
- DigitsNumber to pronounce. The first digit defines how the number should be pronounced:

0—one at a time (example: 411 will be pronounced as four-one-one)

1—date (example: the eleventh of April)

2—time (four eleven AM)

3—phone number (four-one-one)

4—money (four dollars and eleven cents)

5—number (four hundred and eleven)

- User_Ann_ID User Announcement ID (integer) as returned after successful RecordUserAnnouncement request.
- Text ASCII text to pronounce using text-to-speech technology (if supported by the IP equipment).

LANGUAGE	Optional language indicator. Contains a string specifying a language in which the announcement should be made. The valid languages include, but are not limited to, English (US), Spanish, Mandarin, Cantonese, Vietnamese, French, French (Canada), German, Italian, Japanese, Korean, Russian.
----------	--



Play Announcement and Collect Digits

This treatment is used to play an announcement block and collect digits from the caller. Typically, the announcement includes instructions requesting information from the caller.

Parameters: All the parameters of Play Announcement and Collect Digits are recognized.

- Interruptible (check box) indicates whether the caller can interrupt the announcement. IRD instructs URS to send a message to T-Server indicating that the announcement is interruptible (1) or not (0).



Play Application

This treatment is used to execute an application or a script on the IP device. With this treatment you can pass parameters to the application and get return values.

Parameters:

APP_ID	Application ID (integer) specifies the application to be run.
LANGUAGE	Language specifier (string). Valid only for Stentor Network T-Server.
<any number>	You can specify any number of custom parameters by clicking the icon to add an item () in the properties dialog box.

This object lets you indicate the custom parameter type (string or numeric). Specify the type of parameter by using a prefix in the parameter name. Allowed prefixes: {s} and {d}. URS will cut off the prefix from the parameter name and use it as indicator of type: s for string and d for digits. [Figure 214](#) shows an example.

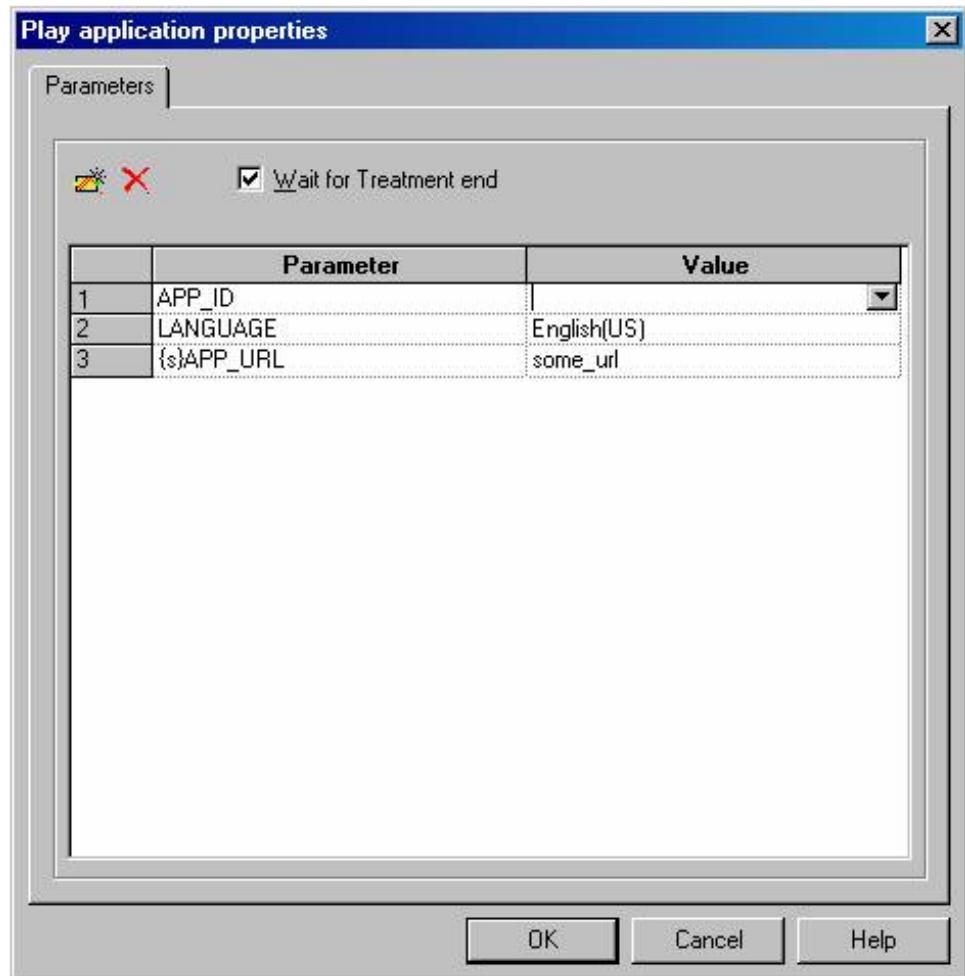


Figure 214: Play Application Properties



RAN stands for “Recorded ANnouncement.” Similar to Music; its source must be a RAN Port on the switch. Only a limited number of switches support this treatment.

Parameters:

ROUTE

DURATION The time, in seconds, that the treatment is applied. Duration is used because it is not always possible to notify URS that a treatment is complete.



Record User Announcement

This treatment is used to create a user-specific announcement. The treatment device returns an announcement ID for a newly created announcement, which the application can use later to trigger playback of the user-specific announcement.

Depending on whether you select or leave clear the Interruptible check box, IRD instructs URS to send a message to T-Server indicating that the announcement is interruptible (1) or not (0).

Parameters:

PROMPT	Contains sublists specifying an initial announcement block to be played. The general format is the same as Play Announcement. Additional restrictions can be imposed by specific implementations.
USER_ID	User ID string specifying the user for whom the announcement will be recorded.
ABORT_DIGITS	The sequence of up to two keys that the caller can enter to abort the recording process. The IP is to consider this as a failed recording attempt.
TERM_DIGITS	The sequence of up to two keys that the caller can enter to indicate that the caller has finished recording the announcement.
RESET_DIGITS	The sequence of up to two keys that the caller can enter to restart the recording announcement. Any announcement recorded up to the point of these keystrokes will be discarded.
START_TIMEOUT	The number of seconds the IP should wait for the callers to begin recording their announcements.
TOTAL_TIMEOUT	The total number of seconds the IP should wait for the callers to finish recording their announcements.



Ringback

This treatment is used to connect the call to a Ringback Tone source. The caller hears the signal of a ringing call.

Parameter for Compatible Mode:

DURATION The time, in seconds, that the treatment is applied.



Set Default Destination

This treatment sets the default routing destination.

Parameters:

DESTINATION Default routing destination (string).



Silence

This treatment is used to connect the call to a source of silence. The caller does not hear anything during the corresponding interval.

Parameter for Compatible Mode:

DURATION The time, in seconds, that the treatment is applied.



Text-To-Speech

This treatment is essentially the same as Play Announcement, except all announcement elements are of type *text*. This option does not allow mixing recorded announcements with text-to-speech. It is used when Play Announcement is not supported.

Parameters:

LANGUAGE Optional language indicator. Contains a string specifying a language in which the announcement should be made.

PROMPT Contains a number of sublists (from 1 to 10). Each sublist is named with a number ranging from 1 to 10 and contains a number of entries describing announcement elements, which include Interruptible and Text.

INTERRUPTIBLE

Indicates whether the caller can interrupt the announcement. Depending on whether you select or leave clear the

Interruptible check box, IRD instructs URS to send a message to T-Server indicating that the announcement is interruptible (1) or not (0).

TEXT ASCII text to pronounce using text-to-speech technology.



Text-To-Speech and Collect Digits

This type of action request generates speech from the text and then collects digits. Typically the speech request includes a request for caller information.

Depending on whether you select or leave clear the Interruptible check box, IRD instructs URS to send a message to T-Server indicating that the announcement is interruptible (1) or not (0).

Parameters: All of the Text-to-Speech parameters are recognized.



Verify Digits

This treatment is used to prompt a calling party to enter digits that will be compared with a desired response.

Parameters:

The general format for the following five sublists is the same as in Play Announcement.

PROMPT	Contains sublists specifying an initial announcement block to be played.
REPROMPT	Contains sublists specifying an announcement block to play after verification has failed and asks caller to reenter information.
SUCCESS	Contains sublists specifying a success announcement.
FAILURE	Contains sublists specifying a failure announcement.
TIMEOUT	Contains sublists specifying a no-input timeout announcement.
LANGUAGE	Optional language indicator. Contains a string specifying the language for an announcement.

The following three **COMPARE** parameters are mutually exclusive:

COMPARE_DIGITS	Contains the actual digits the caller input should be compared against.
COMPARE_USER_ID	Contains the user ID string that IP should use to index into a local table for verification.



COMPARE_PLAN_ID

Contains an index into an IP table of dialing plans. The input is checked for general format compliance with the dialing plan selected by the **COMPARE_PLAN_ID**.

NUM_ATTEMPTS

Number of attempts the caller is allowed to make before failing the verification.

TIMEOUT

Number of unsuccessful attempts to collect input from the caller before timeout.

NUM_DIGITS

Number of digits to be collected. The maximum number of digits that can be collected is 31. The maximum number of digits can be equal to 0. In this case, no time is spent waiting for the caller to input digits, and a response is returned indicating 0 digits collected. The standard does not specify whether the response should contain a success or a failure indication, so expect an undefined behavior.

ABORT_DIGITS

This sequence of up to two keys aborts the digit-collection operation. If this sequence appears, IP considers this as a failed digit-collection attempt.

IGNORE_DIGITS

This sequence of up to two keys is treated as though they have not been pressed.

BACKSPACE_DIGITS

This sequence of up to two keys discards the previous keystroke.

TERM_DIGITS

This sequence of up to two keys causes all the digits, not including the **TERM_DIGITS**, to be returned to the service logic as collected digits.

RESET_DIGITS

This sequence of up to two keys causes all the previous keystrokes to be disregarded. The digit collection resumes.

CLEAR_FLAG

Clear flag indicates whether any information that has been input will be cleared before digit collection starts. Not supported in GR-1129-CORE protocol implementation.

START_TIMEOUT

The number of seconds the IP should wait for the caller to begin DTMF input.

DIGIT_TIMEOUT

The number of seconds the IP should wait between DTMF digits.

TOTAL_TIMEOUT

The total number of seconds the IP should wait for the caller to provide the requested DTMF input.

Access Control

Note: Starting with 7.6, the `Environment` and `Tenant` objects have an option that defines the access permissions for a new `Person` object created in Configuration Manager. The default is for a new `Person` object to have no access permissions. In order for a user to log into the IRD application, the associated `Person` object must have the appropriate access permission. For more information, see the *Genesys 8.1 Security Deployment Guide*. The remainder of this section discusses access control associated with IRD objects.

To safeguard strategies, especially those used in production, you can set access control or permissions for every IRD object, strategy, and subroutine. Users (`Persons`) can be grouped according to roles and provided access permissions to the objects accordingly. Users can also be assigned more than one access level.

You can set access permissions for all objects in Configuration Server, including strategies, agents, agent groups, access groups, DNs, and so on. Access permission is set up in Configuration Manager. You can choose from the following permission settings:

- `No Access`—no permission to read or change this object
- `Read`—permission to read information and receive updates about this object
- `Change`—permission to Read, Change, Execute, and Delete
- `FULL CONTROL`—all permissions for this object
- `Special Access`—any user-defined combination of permissions, including `Read`, `Create`, `Change`, `Execute`, and `Delete`; `Read Permissions`; and `Change Permissions`.

Figure 215 shows the access permissions assigned in Configuration Manager to `Person 4555` in the `Esther` tenant.

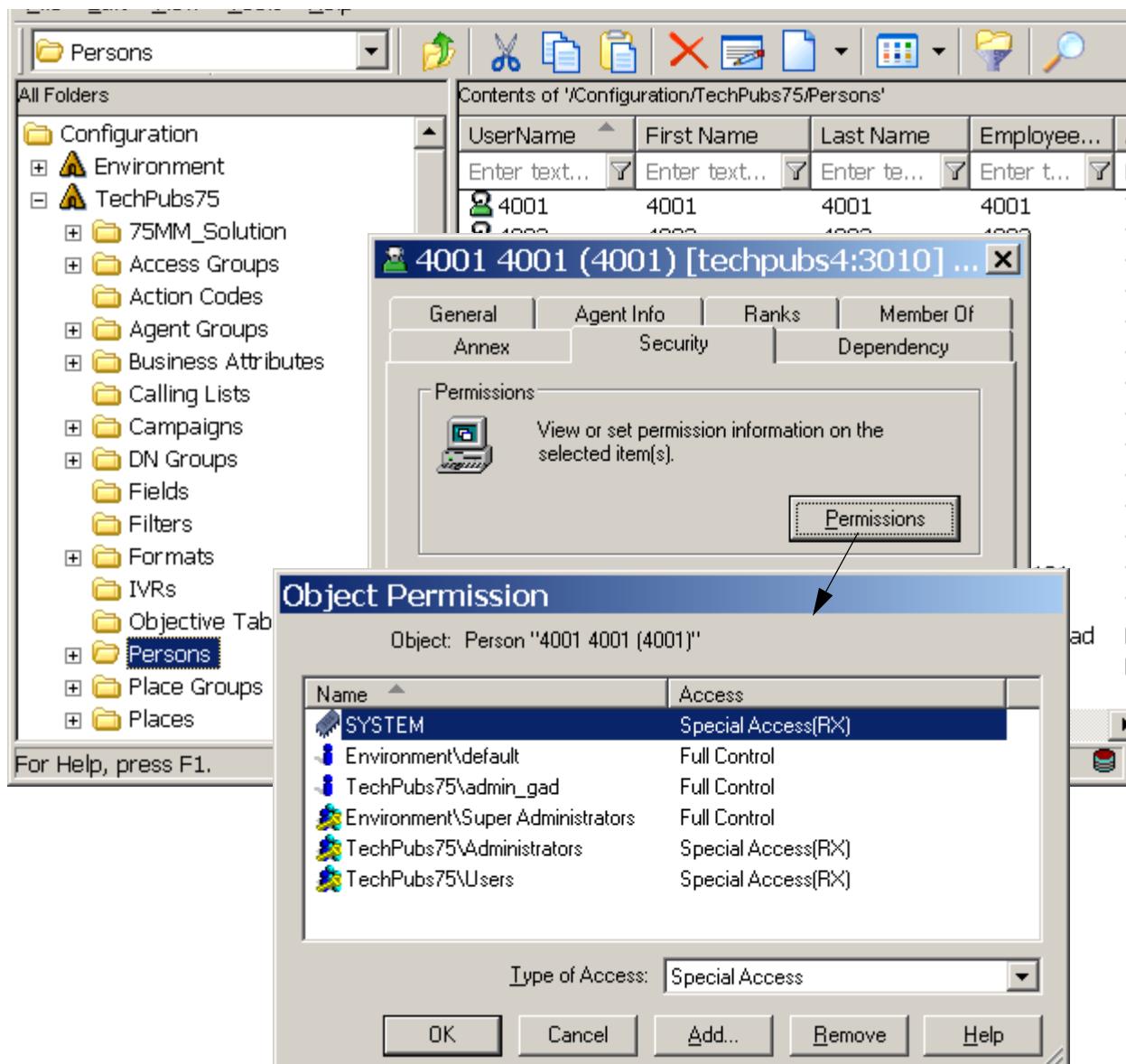


Figure 215: Configuration Manager, Persons Object Permissions

When users launch IRD and log in, they are granted access to IRD strategies, subroutines, and objects based on the group(s) in which they are members. The access permission for strategies, subroutines, and objects appear in their respective List views in IRD. (Access permissions in the List view use different naming conventions than appear in Configuration Manager.) See the *Universal Routing 8.1 Interaction Routing Designer Help* for information on setting up access permissions for objects.

Note: If you add a user to an access group with a specified access level, when the user launches Interaction Routing Designer and logs in, the user can access strategy-building objects based on the level set for the group. If the user is deleted from the group while the user is still logged in, the IRD screen goes blank. When the user is added back to the access group, the IRD screen is restored.

Access Control Example

You decide to create three access groups, Users, Designers, and Administrators, each with different access permissions. *Users* can only view strategies and monitor statistics provided by URS. *Designers* can create strategies and subroutines but cannot change routing rules used in production. *Administrators* have the highest access and can create and change all strategies and routing rules, including those used in production. None of these groups can change or delete routing targets. [Table 127](#) illustrates this example.

Table 127: Example of Access Control Usage

Permission Levels					
Group	Strategy Script Object	Subroutine Script Object	Transaction Objects	Routing Points	Routing Targets (DN, Agent, Place, Agent Group)
User	Read	Read	Read	Read	Read
Designer	Create, Change (limited—cannot change strategies used in production), Read	Create, Change (limited—cannot change subroutines used in production), Read	Create, Change (limited—cannot change routing rules used in production), Read	Read	Read
Administrator	Create, Change, Read	Create, Change, Read	Create, Change, Read	Change, Read	Read

To set up the three groups, create three Access Groups in Configuration Manager and configure the appropriate permission level for each group. Then set up permissions for each object used in the strategy.

As previously noted, Designers can create new strategies, subroutines, and transaction objects and can change any strategies, subroutines, and transaction objects that are not used in production.



To do this:

1. Create a subfolder in the Configuration Manager Scripts folder for production strategies and set the folder's permission level to Read.
2. Place all Strategy Script objects for strategies in production in this subfolder.

All Strategy Script objects placed in this folder, regardless of their previous permission levels, are now set to Read.
3. Similarly, create additional subfolders for the Subroutines Script objects and the routing rules Transaction objects that are used in production and set their permissions to Read.

Now when Designers log in, they can only *read* the strategies used in production. They can *edit* all other existing strategies or create new strategies. Designers can use routing rules that are used in production but cannot change them.

For more information about access permissions, see the *Framework 8.1 Configuration Manager Help*. For instructions on setting permissions for the objects, see *Interaction Routing Designer 8.1 Help*.

Access Control and Sharing Strategies

If you intend to share strategies among users or package strategies to provide to users in a different configuration environment or tenant, be aware that the permission level set for each strategy object is also part of the information stored for the object.

Users without the appropriate permission level cannot unpackage the strategy objects.



Chapter

3

Interaction Routing Designer Functions

Unless noted otherwise, the functions described in this chapter are listed in the [IRD Function Properties](#) dialog box (see [Figure 71](#) on [page 134](#)). The [Function Properties](#) dialog box opens when you place a Function object in your strategy and then double-click it.

Use the [Function Properties](#) dialog box to select a function and enter values for the parameters of that function. For general rules on using the Function object, see [page 127](#). Functions can also be accessed within an expression (see [page 87](#)).

Note: In most cases, only a few of these functions are needed to build a particular routing strategy and not every strategy uses functions. It is not necessary to be familiar with all the functions to use IRD successfully.

The topics in this chapter include:

- [Value Types, page 424](#)
- [Summary of All Functions, page 430](#)
- [Specialized Functions, page 449](#)
- [CallInfo Functions, page 450](#)
- [Configuration Options Functions, page 469](#)
- [Data Manipulation Functions, page 480](#)
- [Date and Time Functions, page 485](#)
- [Force Functions, page 492](#)
- [List Manipulation Functions, page 494](#)
- [Miscellaneous Functions, page 505](#)
- [Reporting Functions, page 556](#)
- [Statistical Functions, page 558](#)

- [String Manipulation Functions, page 581](#)
- [Target Manipulation Functions, page 588](#)
- [Retired Functions, page 616](#)

Note: The IRD installation process places an `r1u.zcf` file in the IRD installation directory. After you import the `r1u.zcf` file into IRD, you get sample utility subroutines. For information on these utility subroutines, see the Samples chapter in the *Universal Routing 8.1 Deployment Guide*.

Value Types

Each function description in this chapter includes a returned value type. The types `STRING`, `FLOAT`, and `INTEGER` refer to character strings, floating-point numbers, and integer numbers respectively. `VOID` is used as a return value type to indicate that a function returns no value. `STRING` (list) and `STRING` (target) indicate special values of type `STRING`; their formats are discussed below. Special value types include `TIME`, `DAY`, and `DATE`.

Important Information

- All the parameters of a function (including optional parameters) must receive values, even though the values of one or more parameters can sometimes be ignored by the function. For the optional parameters of the functions that you do not specify or for parameters of the functions that are not objects being used by Configuration Layer, IRD automatically inserts empty strings (two single quotation marks only).
- When using string constants in a strategy, the number of characters that make up the constant (what appears between the quotes) is limited to 1000 characters. For example, `Assign[x, '...over 1000 characters']` is not supported.

Type Conversion

In cases in which the strategy compiler detects a value types mismatch, it implicitly inserts type conversion code into the strategy. Type mismatching can happen:

- When a function is used and the type of a supplied value for one of the parameters is different from what the function expects. The conversion code will convert the type of supplied value into the type of the functional parameter.
- When assignment is used and the type of calculated value is different from the type of variable. The conversion code will convert the type of the calculated value into the variable type.

- When the Expression Builder dialog box (see Figure 39 on [page 88](#)) is used where values of different types are compared. The conversion code will convert the type of one of parameters into type of another one. If conversion is possible in both directions, then the second parameter will be converted to the type of first one.

[Table 128](#) summarizes the properties of values types.

Table 128: Value Type Properties

Value type	Native (internal) presentation	Constants of this type
INTEGER	Computer dependent. Maps to C programming language type long. Usually has the size of 4 bytes and a range of values from -2147483648 to 2147483647	Presented in strategy as a sequence of digits. Can be prefixed with a plus or minus sign. Should not exceed allowed range. Samples: 123, 0, -234.
FLOAT	Computer-dependent. Maps to C programming language type double. Usually has the size of 8 bytes and a range of +/-1.7E308 with at least 15 digits of precision.	Generic format [+/-]ddd.ddd[eddd] where optional parts are taken in [].ddd presents any not-empty sequence of digits. Samples: 123.0, 0.5, 234.111, 5.5e6.
STRING	Zero-terminated sequence of bytes.	Generic format for any sequence of characters in quotes. Cannot contain the quote character inside. Should not exceed 1000 characters. Samples: 'abcdefg', ''.
LIST	KVList data object as supported by Genesys Common library	No constants of this type.
DAY	Integer value from 0 to 6. Sunday presented as 0, Monday as 1, and so on.	Presented in strategy with a number from 0 to 6 or with special names that represent these numbers: Sunday, Monday ... Saturday. These names are not strings so do not include them in quotes. Samples: Sunday, 1, 6.
DATE	Integer value presenting date in number of days from January 1, 1970.	Generic format mm/dd/yyyy. Year part can be missed; the current year is assumed in such case. They are not strings so do not include them in quotes. Samples: 10/29/2007, 10/25.
TIME	Integer value presenting time of day in number of minutes from midnight 00:00.	Generic format hh:mm [AM/PM]. They are not strings so do not include them in quotes. Samples: 5:09 PM, 21:00.

Conversion Rules

[Table 129](#) contains conversion rules for the various value types.

Table 129: Conversion Rules

Type Converted To	Result
Conversion rules for INTEGER type results in following values if converted to:	
FLOAT:	The same value in float format. For example, 123 -> 123.0.
STRING:	String presenting the integer value. For example, 123 -> '123'.
DAY:	Remainder of dividing the Max(0, value) by 7.
DATE:	Max(0, value).
TIME:	Remainder of dividing the Max(0, value) by 1440.
Conversion rules for FLOAT type results in following values if converted to:	
INTEGER:	Fractional part is truncated. For example 123.8 -> 123.
STRING:	String presenting the float value. For example 123.8 -> '123.8'.
DAY:	Not allowed, compiler generates error.
DATE:	Not allowed, compiler generates error.
TIME:	Not allowed, compiler generates error.
Conversion rules for STRING type results in following values if converted to:	
INTEGER:	Integer value presented by string fragment from the beginning of string to the first character that cannot be interpreted as part of the integer value. For example, '503abc' results in 503. If the string fragment is empty, then 0 will be used. Plus or minus sign as first character of integer value is also allowed. If the string fragment is too long (exceeds allowed range for INTEGERS), the result is unpredictable.
FLOAT:	Float value presented by string fragment from the beginning of the string to the first character that cannot be interpreted as part of float value. For example, '+503.12e6abc' results in 503120000. If the string fragment is empty, then 0 will be used. Plus or minus sign as the first character of the float value is also allowed. If the string fragment is too long (exceeds allowed precision for FLOAT), then out-of-range digits are replaced with 0.

Table 129: Conversion Rules (Continued)

Type Converted To	Result
LIST:	If STRING has Key-Value List or JSON format, then the resulting LIST object will contain keys and values from this string. If STRING doesn't have format of Key-Value List then the resulting LIST object will be empty.
DAY:	The same rule as for conversion to INTEGER. Can potentially result in invalid DAYS if the resulted INTEGER is out of range 0-6.
DATE:	Number of days from 01/01/1970 if the string has a format of 'mm/dd/yyyy', 'mm/dd/yy', 'mm/dd'. If the year is missed, then the current year is assumed. The result is unpredictable if the string has a different format.
TIME:	Number of minutes from midnight if the string has a format of 'hh:mm', 'hh:mm PM', or 'hh:mm AM'. A format with hh:mm:ss is also allowed, but seconds are ignored. The result is unpredictable if the string has a different format.
Conversion rules for LIST type results in following values if converted to:	
INTEGER, FLOAT	Not allowed, compiler generates error
STRING:	LIST value will be converted to STRING in format of Key-Value List (example: 'aaa:5 gggg:123 ...') with keys and values taken from LIST object
Conversion rules for DAY type results in following values if converted to:	
INTEGER:	Internal presentation of DAY type (0-6 integer value)
FLOAT:	Not allowed, compiler generates error
STRING:	The same as for INTEGER but presented as string
DATE:	Not allowed, compiler generates error
TIME:	Not allowed, compiler generates error
Conversion rules for DATE type results in following values if converted to:	
INTEGER:	Internal presentation of DATE type
FLOAT:	Not allowed, compiler generates error
STRING:	String in format "mm/dd/yyyy"
DAY:	Day of week the provided DATE is.

Table 129: Conversion Rules (Continued)

Type Converted To	Result
TIME:	Not allowed, compiler generates error
Conversion rules for TIME type results in following values if converted to:	
INTEGER:	Internal presentation of DATE type
FLOAT:	Not allowed, compiler generates error
STRING:	String in format 'hh:mm'
DAY:	Not allowed, compiler generates error
DATE:	Not allowed, compiler generates error

Key-Value List

The key-value list is a data structure that consists of multiple data entries (values) each of which is associated to some string constant (key). The keys allow access to the values. Several values can have the same key.

URS operates with string representations of key-value lists. The format of the string representation of a key-value list is the following:

Key1:Value1|...|KeyN:ValueN

Here, Key1, ..., KeyN, Value1, ..., and ValueN are all strings without enclosing quotation marks. Any spaces are interpreted as part of the corresponding key or value.

The keys can be composite: in the above notation, Key1 can have the form of KeyI1.KeyI2....KeyIJ. This feature makes it possible to operate with nested key-value lists.

Everywhere in this chapter, the type STRING (list) denotes values of type STRING that have this format.

Warning! When creating a key-value list that uses a string format (example: aaa:5|gggg:123|...), the maximum Key length cannot exceed 1000 bytes and the maximum length of one key-value pair cannot exceed 4096 bytes.

STRING (Target)

The STRING (target) type denotes a STRING (list) type value with predefined keys, which differ for the various functions. Wherever this type occurs in the description of a function, the relevant keys are enumerated and explained.

STRING (Statistical Object)

The STRING (Statistical object) type denotes a STRING type value that specifies a statistical object. It has the format <Name>@<statserver>. <type>.

Note: Because the name of a DN target type can include the symbol @, the first @ that appears in the specification of this target type is always considered to be a part of its name, and not a separator between the name and the location. For example:

The string *name1@name2.A* denotes a target of the Agent type with the name *name1*, and a location of *name2*.

The string *name1@name2.DN* denotes target of the non-configured DN type with a name of *name1@name2* and a location that is not specified (the location is then taken from that set in the URS Application object's Options tab).

DN Target Format

The format for specifying target DNs in functions is <number>@<switch>.DN. This differs from the format used in the Percentage routing rule, which uses the format <number>@<switch>.

Quote Usage, Variables, and Function Parameters

For many functions, you can specify parameters as a variable or a constant (STRING or INTEGER). Variables must first be defined in the Variable list in order to use them in a function. The value entered is first checked against all defined variables in the strategy. If IRD finds a variable with the name specified for the value, then the value is considered a variable.

If no variable is found, the value is checked for the symbols [and]. If found, the value is considered to be a function call. If the entered value does not contain these symbols, the value is considered to be a constant, and when you click the Add button, the constant is automatically placed in single quotes (if it is not in single quotes already) in the Expression section of the function's dialog box.

See “Variable Usage and Strategy Design” on [page 92](#).

Warning! You do not need to enter quotation marks manually in the Expression section of the function's dialog box. Quotation marks are automatically added when you click the Add button. Do not use quotes in variable names or string constants.

Specifying Variables in Dialog Boxes

When using a variable, select the variable from the drop-down list or type the variable name manually.

- For parameters that accept strings or variables: when a variable is entered manually, IRD searches for the variable from the Variable list and selects the one with the same name. If the variable is not found, IRD interprets the variable as a string.
- For parameters that accept integers or variable: variables cannot be entered manually only selected from the drop-down list.

Virtual Queues and DNIS Information

Every Virtual Queue event that URS distributes will contain the current DNIS for the interaction. This means that URS will not only check incoming T-Server events (`EventRouteRequest`) but also outgoing events (`EventRouteUsed`) to keep the DNIS value up to date.

Summary of All Functions

Note: [Page 616](#) lists retired functions.

[Table 130](#) lists all the functions available for use in strategies, a function description, the IRD category in which they fall, and the page number at which the function information can be found. The table contains brief descriptions of the functions, for quick reference.

Warning! To avoid problems, you are strongly advised to read the detailed description of the function later in this chapter.

Table 130: Interaction Routing Designer Functions

Name	Brief Description	Category	Page
ACDQ	The value of <code>ThisQueue</code> in the Call Information attributes that came with the event that started the strategy.	CallInfo	451
acfgdata	See note on page 559 .		
ActiveServerName	Returns the name of the active T-Server from the pair working in Hot Standby or Warm Standby mode.	Miscellaneous	505

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
Alarm	Sends an alarm message through the URS management API.	Miscellaneous	506
ANI	Automatic Number Identification; sent as an attribute of the event that started the strategy.	CallInfo	451
AnswerCall	Attempts to answer the current interaction.	Miscellaneous	506
Attach	Attaches User Data to the interaction.	CallInfo	451
BearerCapability	Specific to some Network T-Servers. Returns the type of device used to make the interaction.	CallInfo	453
BlockDN	Blocks and unblocks a specific DN for specified period of time.	Target Manipulation	589
BusinessData BusinessDataINT	Retrieves the User Data information collected from the incoming call, when the caller responds to either the IVR or to an agent. BusinessData returns a string; BusinessDataINT returns an integer.	CallInfo	453 454
Bytes	This function returns the string, presenting space-separated numerical values of every character from the input string.	String Manipulation	581
callage	See note on page 559 .		
CallID	The interaction identification provided by the switch, which uniquely identifies the interaction; sent as an attribute of the event that started the strategy.	CallInfo	454
CallsDistributed	Returns the total number of interactions distributed by URS from all routing points, from the current routing point, or from a given virtual queue.	Statistical	560

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
CallsEntered	Returns the total number of interactions that URS has started processing from all routing points or from the current routing point, or the total number of interactions that have entered a virtual queue.	Statistical	560
CallsWaiting	Returns the number of interactions currently waiting for a given target.	Statistical	561
CallType	Returns an integer denoting the type of the interaction (Unknown, Inbound, Outbound, Internal, or Consult).	CallInfo	454
CallUUID	Returns attribute CallUUID.	CallInfo	455
Cat	Concatenates two strings.	String Manipulation	582
CCTExtractTargets	Creates a list of targets from a key-value list in a specific format and stores data for later use in translation.	Target Manipulation	590
CED	Returns the sequence of digits entered by the caller during appropriate treatments.	CallInfo	455
CheckAgentState	Instructs URS on whether to take into account the Stat Server report on agent availability.	Miscellaneous	507
ClaimAgentsFromOCS	Notifies the specified Outbound Contact Server (OCS) to free some resource for processing inbound interactions.	Miscellaneous	508
ClearTargets	Removes the interaction from all virtual queues or from a specified virtual queue.	Miscellaneous	510
ClearThresholds	Invalidates all thresholds previously set by threshold objects.	Statistical	562
ClearUpdateTrigger	Remove the specified trigger key from this list of keys.	Miscellaneous	511

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
ConnID	The connection identifier assigned to the interaction by T-Server; sent as an attribute of the event that started the strategy.	CallInfo	455
CountSkillInGroupEx	Returns the number of agents in an Agent Group that satisfy a given skill condition.	Miscellaneous	511
CountTargetsByThreshold	Returns the number of targets from the Targets List parameter (possibly 0).	Miscellaneous	514
CreateSkillGroup	Converts the provided Agent Group, Skill Expression, and Stat Server into a target while keeping the connection between target source and target itself.	Miscellaneous	514
Date	Returns the current date in the MM/DD/YYYY format.	Date/Time	486
DateInZone	Returns the current date in the specified time zone.	Date/Time	486
Day	Returns the current day of the week.	Date/Time	486
DayInZone	Returns the current day of the week in the specified time zone.	Date/Time	487
Delay	Pauses the execution of the strategy for an interval specified in milliseconds.	Miscellaneous	515
DeleteAttachedData	Deletes interaction User Data.	CallInfo	456
DeliverCall	Attempts to deliver the interaction to a DN on the basis of low-level target information or sends the interaction back from such a DN to the routing point from which it entered.	Target Manipulation	591
DeliverToIVR	Attempts to deliver the interaction to a DN on the basis of high-level target information, and keeps trying until it succeeds or a specified timeout elapses.	Target Manipulation	593

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
Dest	The value of ThisDN in the Call Information attributes that came with the event that started the strategy. Generally, this is the value of the DN at which the interaction that started the strategy arrived.	CallInfo	456
DistributedPercentage	Returns the percentage of distributed interactions among the last 100 interactions processed by URS from all routing points or from the current routing point, or the percentage of interactions distributed from a virtual queue among the last 100 interactions that left the virtual queue.	Statistical	562
DistributedWaitingTime	Returns the average waiting time of the interactions distributed from all routing points, from the current routing point, from a virtual queue among the last 100 interactions processed from all routing points, or from the current one or diverted from the specified virtual queue.	Statistical	563
DNIS	Dialed Number Identification Service; sent as an attribute of the event that started the strategy.	CallInfo	456
ExcludeAgents	Instructs URS not to route interactions to any agent on the specified list of agents	Configuration Options	470
ExpandGroup	Returns the list of Agents or Place targets belonging to a specified Agent Group or Place Group target in high-level target format.	Miscellaneous	515
ExpandWFActivity	Returns list of agents scheduled for an activity.	Miscellaneous	517
ExtensionAttach	Adds a specified key-value pair to the Extensions list of the interaction.	CallInfo	456

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
ExtensionData	Retrieves the value of a given key from the Extensions list of the interaction.	CallInfo	457
ExtensionUpdate	Adds a specified key-value pair to the Extensions list of the interaction and removes any previously present pairs with the same key.	CallInfo	457
ExtrouterError	Use for external routing to change the default URS reaction in the case of a failure to get a remote access number.	Miscellaneous	518
ExtrouterStatus	Returns 1 if external routing is possible and <0 if not. The parameter for this function is a remote location to which the interaction is being routed.	Miscellaneous	518
FindConfigObject	Returns information about a requested configuration object.	Configuration Options	471
FindServiceObjective	Finds the service objective for a given combination of customer segment, service type, and media type.	Data Manipulation	480
FirstHomeLocation	Returns the value in the <code>FirstTransferHomeLocation</code> field in the T-Server event that started the strategy.	CallInfo	457
Force	Instructs URS to force-route the interaction to a target encountered in the first target object. The target is interpreted as specifying a DN and the switch of that DN, an Agent, or Place. The interaction is definitively routed without checking target availability.	Force	493
GetAvgStatData	Calculates the specified statistic for all listed targets and returns the average value of this statistic.	Statistical	564
GetConfigOption	Retrieves the current value of a specified URS configuration option.	Configuration Options	472

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
GetCurrentLeg	Returns information about the DN where the interaction is located to which the function is called.	CallInfo	457
GetCurrentScript	Retrieves the current value of a specified URS strategy name.	Configuration Options	473
GetCurrentSwitch	Retrieves the name of the switch to which the interaction is currently located.	CallInfo	458
GetCurrentTServer	Retrieves the name of the T-Server application controlling the switch to which the interaction is currently located.	CallInfo	458
GetCustomerSegment	Gets the attached data that has the CustomerSegment key used for routing decisions and report generation.	CallInfo	459
GetIntegerKey	Retrieves the integer value associated to the first occurrence of a specified key in a key-value list.	List Manipulation	495
GetLastErrorInfo	Returns the error code returned by some T-Server as a concatenation of the error number and the error message.	Miscellaneous	518
GetMaxStatData	Calculates the specified statistic for all listed targets and returns the maximum value of this statistic.	Statistical	564
GetMaxSubList	Parses out key-value pair(s) for pairs with the maximum value.	List Manipulation	495
GetMediaType	Returns an integer assigned to the interaction by T-Server.	CallInfo	459
GetMediaTypeName	Returns the name of the specified media type.	Configuration Options	474
GetMinStatData	Calculates specified statistic for all listed targets and returns the minimum value of this statistic.	Statistical	565

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
GetMinSubList	Parses out key-value pair(s) for pairs with the minimum value.	List Manipulation	495
GetObjectProperty	Returns the value of the requested option of the configuration object identified by Type, Subtype and Name.	Configuration Options	473
GetPriority	Returns the current global priority of the interaction.	Miscellaneous	519
GetRawAttribute	Returns the value of a specific attribute (or empty string if there is no such attribute) when IRD queries attributes that are allocated to a TEvent.	List Manipulation	496
GetRemoteAccessCode	Requests an access code from a remote T-Server for a target specified in low-level format, and records the received access code in the target.	Target Manipulation	594
GetRoutingPoint	Retrieves the routing point from which URS started routing the current interaction.	CallInfo	460
GetServiceObjective	Gets the time objective to service an interaction.	CallInfo	460
GetServiceType	Gets the type of service being requested by the caller.	CallInfo	460
GetSkillInGroupEx	Returns a list of all agents in an Agent Group that satisfy a given skill condition. The list is returned in high-level target format.	Miscellaneous	519
GetStringKey	Retrieves the string value associated to the first occurrence of a specified key in a key-value list.	List Manipulation	495
GetUTC	Returns universal coordinated time in seconds.	Date/Time	487

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
IncrementPriority	Extended version of Increment function, which has been retired. Increments priority for specified value in specified timeout.	Target Manipulation	596
IncrementPriorityEx	Provides the capability to increment priority starting after a specified time interval.	Target Manipulation	596
InformationDigits	Specific to some Network T-Servers. Returns a two-digit combination encoding specific information about the origin of the interaction.	CallInfo	461
InteractionData InteractionDataINT	Retrieves the User Data collected from the incoming call. InteractionData returns a string; InteractionDataINT returns an integer.	CallInfo	461, 462
InVQWaitTime	Provides expected wait time for a call in a virtual queue.	Statistical	565
IsSpecialDay	Checks whether the current date matches any of the Statistical Days in a configured Statistical Table, or whether the current date matches a specified Statistical Day.	Date/Time	488
IsSpecialDayEx	Same as above, but provides time zone and the option to use or not use time limits with a Statistical day.	Date/Time	488
JumpToStrategy	Starts a strategy saved as a script in Configuration Layer; the name of the script is passed as a parameter.	Miscellaneous	522
JumpToTenant	Behaves like JumpToStrategy, but includes an additional parameter Tenant Name representing the name of tenant	Miscellaneous	523
KeepQueue	Prevents URS from distributing any call with priority less than current one if called with the value of type Constant set to true.	Target Manipulation	597

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
KVListFindSubList	Presents an extension of an existing set of KVList functions. It finds and returns the first sublist inside the input list that satisfies the search criteria.	List Manipulation	497
KVListGetKey	Returns the name of the key in a key-value pair	List Manipulation	497
KVListGetValue	Returns a sublist for a List that has a key-value pair with name Key.	List Manipulation	498
KVListGetSize	Returns the number of top-level key-value pairs in the List.	List Manipulation	498
KVListGetStringValue	Returns the value of a key-value pair with name Key in string format.	List Manipulation	498
LATA	Specific to some Network T-Servers. Returns the LATA (Local Access Transport Area) of the interaction's origination point.	CallInfo	462
lcfgdata	See note on page 559 .		
ListGetDataCfg	Returns the specified property of the IRD List object (see page 73).	Data Manipulation	482
ListGetInteger	Parses out the integer value for the specified key from a key-value list based on the position, rather than the key.	List Manipulation	498
ListGetKey	Parses out the key from a key-value list based on the position rather than the key.	List Manipulation	499
ListGetSize	Returns the number of key-value pairs.	List Manipulation	499
ListGetString	Parses out the string value for the specified key from a key-value list based on the position, rather than the key.	List Manipulation	499

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
ListLookup	Checks whether a given string appears in the result of a database inquiry specified in an Access Table in the Configuration Layer.	Data Manipulation	483
ListLookupCfg	Behaves like ListLookup except that the first parameter is not an Access Table, but an IRD list object (see page 73).	List Manipulation	484
MultiplyTargets	Controls switching into target-multiplication mode.	Target Manipulation	523
MultiSkill	Forms name of Agent Group in the following format: ?:<skillExpression>@<name_of_StatSersver>.GA	Miscellaneous	526
NMTEextractTargets	Retrieves information about interactions sent from the Genesys environment to non-monitored DNs.	Target Manipulation	598
NotDistributedPercentage	Returns the percentage of nondistributed interactions among the last 100 interactions processed by URS from all routing points or the current routing point, or the percentage of interactions diverted but not distributed from a virtual queue.	Statistical	569
NotDistributedWaitingTime	Returns the average waiting time of the last 100 nondistributed interactions from all routing points, from the current routing point, or from a virtual queue.	Statistical	567
NPA	Returns the first three digits of the caller's ANI (in the United States, this is the caller's area code).	CallInfo	462
NPANXX	Returns the first six digits of the caller's ANI (in the United States, this is the caller's area code followed by a prefix).	CallInfo	463

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
OnCallAbandoned	Specifies emergency strategy for reporting purposes when a call is abandoned.	Miscellaneous	527
OnRouteError	Specifies URS reaction to every type of error.	Miscellaneous	527
Orig	The value of <code>OtherDN</code> in the Call Information attributes that came with the event that started the strategy. Generally, this is the value of the DN from which the interaction arrived.	CallInfo	463
OtherTrunk	Returns the value from <code>AttributeOtherTrunk</code> which is taken from the event that starts the strategy	CallInfo	463
PACCode	Same as CED.	CallInfo	463
PACType	Specific to some Network T-Servers. Returns the value of the key <code>PAC_TYPE</code> in the interaction's Extensions list. Consult your T-Server manual about the meaning of this value if the feature is supported.	CallInfo	463
Peg	Creates a peg or increases the value associated with the peg for subsequent interactions. The peg is attached to the interaction with its associated value at the time the interaction is definitively routed.	Reporting	557
PegValue	Same as SData.	Reporting	557
PositionInQueue	Returns the position of the current interaction among the interactions waiting for a specified target.	Statistical	568
Print	Prints a string value into the log file.	Miscellaneous	528
Priority	Sets the global priority of the interaction.	Target Manipulation	602
PriorityLimits	Enables users to define the upper and lower limits of a priority value.	Target Manipulation	603

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
PriorityTuning	Adjust interaction priority by taking into account age, expected wait time, and service objective.	Target Manipulation	603
Rand	Returns a random integer between 1 and max.	Miscellaneous	529
ReleaseCall	Releases the current interaction.	Miscellaneous	529
RequestType	Specific to some Network T-Servers. Returns the value of the key REQUEST_TYPE in the interaction's Extensions list. Consult your T-Server manual about the meaning of this value if the feature is supported.	CallInfo	464
ResetBusyTreatments	Clears the buffer of busy treatments.	Miscellaneous	530
ResetStatAdjustment	Cancels any adjustment that may have been set for a statistic for a given target by the function SetStatAdjustment.	Statistical	569
RouteCall	Attempts to route the interaction to a low-level target.	Target Manipulation	607
Routed	Marks an interaction as routed.	Target Manipulation	607
Router	Returns the name of the URS Application	Configuration Options	474
SData	Returns the current value of a predefined or user-defined statistic for a given target.	Statistical	572
sdata	See note on page 559 .		
SDataInTenant	Returns the current value of a predefined or user-defined statistic for a given target. The statistic is requested in the context of a specific tenant.	Statistical	574

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
SelectDN	Places the interaction on an internal router queue and attempts to select an available target from one or more specified lists and, if it succeeds in the selection, returns low-level information on the selected target.	Target Manipulation	608
SelectTargets	Returns the list of Agent Groups or Place Groups that have not met the quotas listed in a given Quota Table from a list of Agent Groups and Place Groups specified as high-level targets.	Miscellaneous	530
SelectTargetsByThreshold	Finds the best available target(s) from a list of targets by applying a statistic with a threshold comparison against the input target list.	Miscellaneous	530
SendEvent	Provides explicit control to distribute specified events.	Miscellaneous	532
SendRequest	Allows you to send T-Server any possible request message.	Miscellaneous	534
ServerStatus	Reports on whether URS is currently connected to a specified server application.	Miscellaneous	544
SetCallOption	Universal function to have certain options controlled by the strategy instead of URS.	Configuration Options	475
SetDelayedAttach	All invocations of the Attach or Update functions made after the occurrence of a SetDelayedAttach function propagate update events to the T-Server only after a SetDelayedAttach function set to false is called or strategy execution for this interaction is suspended for some reason.	Miscellaneous	545
SetDNIS	Explicitly specifies the DNIS with which the interaction will be routed.	Configuration Options	475
SetDNISOverride	Specifies a way for overwriting the interaction's DNIS before routing it.	Configuration Options	475

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
SetHomeLocation	Allows you to specify the home location for a call.	CallInfo	464
SetIntegerKey	Adds a key-value pair with a prescribed integer value to a specified key-value list, and removes any previously present key-value pairs with the same key.	List Manipulation	500
SetInteractionAge	Overwrites the default age of interaction.	Miscellaneous	545
SetObjectProperty	Allows you to set a value in Annexes of the specified object identified by Type, Subtype, and Name.	Configuration Options	476
SetLastError	Use to flag subroutine errors.	Miscellaneous	546
SetRunTimeMode	Allows the user to specify the run time rules for a specific call.	Miscellaneous	548
SetStatAdjustment	Enforces an adjustment of the values of a specified statistic for a particular target. The statistic is adjusted only when it is used for selecting an available target.	Statistical	574
SetStatUpdate	Periodically updates the value of a statistic without leaving the target selection object.	Statistical	575
SetStringKey	Adds a key-value pair with a prescribed string value to a specified key-value list, and removes any previously present key-value pairs with the same key.	List Manipulation	500
SetTargetThreshold	Defines a statistical threshold for imposing additional readiness conditions for targets. Share Agent by Service Level Agreement routing solutions use this function.	Target Manipulation	612
SetThresholdEx	Defines queue or routing point statistical thresholds for determining the availability of such DNs as targets.	Statistical	577

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
SetTranslationOverride	Specifies an explicit translation table to be used in translating the current interaction instead of a configured access code table.	Configuration Options	476
SetUpdateTrigger	Adds or overwrites a new trigger key in a list of keys.	Miscellaneous	549
SetVQPriority	Sets the priority of the current interaction for the specified virtual queue. Changes the priority of the interaction for all internal router queues composing the virtual queue.	Target Manipulation	613
StateCode	Specific to Network T-Servers. Returns the value of the key REQUEST_TYPE in the interaction's Extensions list.	CallInfo	465
StrAsciiBreak	Returns the position of the first occurrence of one of a set of characters after a specified place in a string.	String Manipulation	582
StrAsciiTok	Parses out the part of a string between two consecutive occurrences of characters from a set of separators. Can be used repeatedly to parse out the elements of a list with any prescribed set of separators. (Also see “ StrNextTokInd ”).	String Manipulation	582
StrChar	Locates an occurrence of a given substring in a given string and returns the index of its first character.	String Manipulation	583
StrFormatTime	This function accepts as input formatting string, the UTC timestamp, and GMT/local timezone selection flag.	Miscellaneous	550
StrGetChar	Retrieves a character by its position in a string.	String Manipulation	584
StrLen	Returns the length of a string.	String Manipulation	584

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
StrNextTokInd	Helps the StrAsciiTok function manage the indexing sequence when using two or more variables.	String Manipulation	585
StrReplace	Replaces characters.	String Manipulation	585
StrStr	Locates an occurrence of a given substring in a given string and returns the tail segment of the original string starting with that substring.	String Manipulation	585
StrSub	Returns a substring identified by its starting and ending positions.	String Manipulation	585
StrTargets	facilitates the creation of a comma-separated list of targets for use as input parameters for the Genesys-provided utility subroutines.	String Manipulation	586
StrToLower	Returns a copy of string with all characters converted to lowercase.	String Manipulation	587
StrToUpper	Returns a copy of string with all characters converted to uppercase.	String Manipulation	588
SuspendForDN	Attempts to select a DN from any of the internal router queues on which the interaction has been placed and keeps trying until either a DN is selected or a specified timeout elapses.	Target Manipulation	614
SuspendForEvent	Synchronizes a strategy's execution with an external event.	Miscellaneous	550
SuspendForTreatmentEnd	Pauses the strategy until the end of the currently playing treatment.	Target Manipulation	615
TargetComponentSelected	Allows reports to be generated on why a particular target object was chosen over other target objects.	Miscellaneous	551
TargetObjectSelected	Returns the target to which the interaction was definitively routed in high-level format.	Miscellaneous	552

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
TargetSelected	Returns the DN and switch to which the interaction was definitively routed.	Miscellaneous	552
TargetSelectionTuning	Causes URS to extend statistic selection using the cost of routing to the specified target as the main criteria.	Target Manipulation	615
TargetState	Accepts standard specified targets and returns all of the information about this target known to URS. (The only supported target types are agent and place. One of these two must be specified.)	List Manipulation	500
ThisTrunk	Returns the value from AttributeThisTrunk which is taken from the event that starts the strategy.	CallInfo	465
Time	Returns the current 24-hour clock time in the hh:mm format.	Date/Time	489
TimeDifference	Produces the difference between two moments specified in milliseconds, assuming that the moments are at most 24 hours apart.	Date/Time	489
TimeInZone	Returns the number of minutes elapsed since the last midnight (00:00 AM) in the specified time zone.	Date/Time	490
Timeout	Sets a timeout interval after which the interaction will be routed to the default destination if it has not been definitively routed already.	Miscellaneous	553
TimeStamp	Returns the current time in milliseconds, measured from midnight the same day.	Date/Time	490
Translate	Performs number translation on a low-level target and returns the translated target.	Target Manipulation	615
TRoute	Explicitly instructs URS to force-route the interaction to a specified target.	Force	493

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
UData	Retrieves the User Data information collected from the incoming interaction when the caller responds to either the IVR or to an agent.	CallInfo	465
UDataINT	Retrieves the User Data information collected from the incoming interaction when the caller responds to either the IVR or to an agent. UDataINT differs from UData because it returns an integer rather a string value.	CallInfo	554
Update	Updates call User Data.	CallInfo	467
UpdateBusinessData	This function updates interaction Business Data.	CallInfo	468
UpdateInteractionData	This function updates interaction User Data.	CallInfo	469
UpdateScript	Attaches information to the User Data structure and clears all existing pairs with the same key as that of the newly attached pair.	Miscellaneous	554
UseActivityType	Allows routing to any agent regardless of campaign assignment based on activity.	Configuration Options	477
UseAgentState	Specifies the agent state URS will use instead of the default one reported by Stat Server.	Miscellaneous	554
UseAgentStatistics	Causes URS to apply statistics for target selection at the level of Agents or Places even if targets are groups of corresponding objects.	Miscellaneous	555
UseCapacity	Instructs URS on the condition for using configured Capacity Tables for computing statistical values.	Statistical	579
UseCustomAgentType	Controls whether calls can be routed to a specific Agent.	Configuration Options	478

Table 130: Interaction Routing Designer Functions (Continued)

Name	Brief Description	Category	Page
UseCustomDNTYPE	Controls whether calls can be routed to a specific DN.	Configuration Options	478
UseCustomPlaceType	Controls whether calls can be routed to a specific Place.	Configuration Options	478
UseDNTYPE	Specifies the type of DN to which calls or other interactions should be sent.	Configuration Options	479
UseMediaType	Specifies the media types associated with a target.	Configuration Options	479
UTCAdd	Increments the provided universal coordinated time with the specified number of years, months, and so on.	Date/Time	490
UTCFromString	Takes a string in universal coordinated time format and returns the number of seconds.	Date/Time	491
UTCToString	Takes universal coordinated time number of seconds and converts it to a string.	Date/Time	492
VQSelected	Returns the alias of the virtual queue to which the interaction was definitively routed.	Miscellaneous	555

Specialized Functions

The functions BearerCapability ([page 453](#)), InformationDigits ([page 461](#)), LATA ([page 462](#)), PACType ([page 463](#)), RequestType ([page 464](#)), and StateCode ([page 465](#)) are designed to retrieve special interaction information, which is only present in the interaction data for a small number of Network T-Servers. Consult T-Server documentation on whether these functions can be used meaningfully.

The SetCallOption function ([page 475](#)) enables the strategy to control options instead of URS.

CallInfo Functions

The IRD Function Properties dialog box with CallInfo selected is shown in Figure 216.

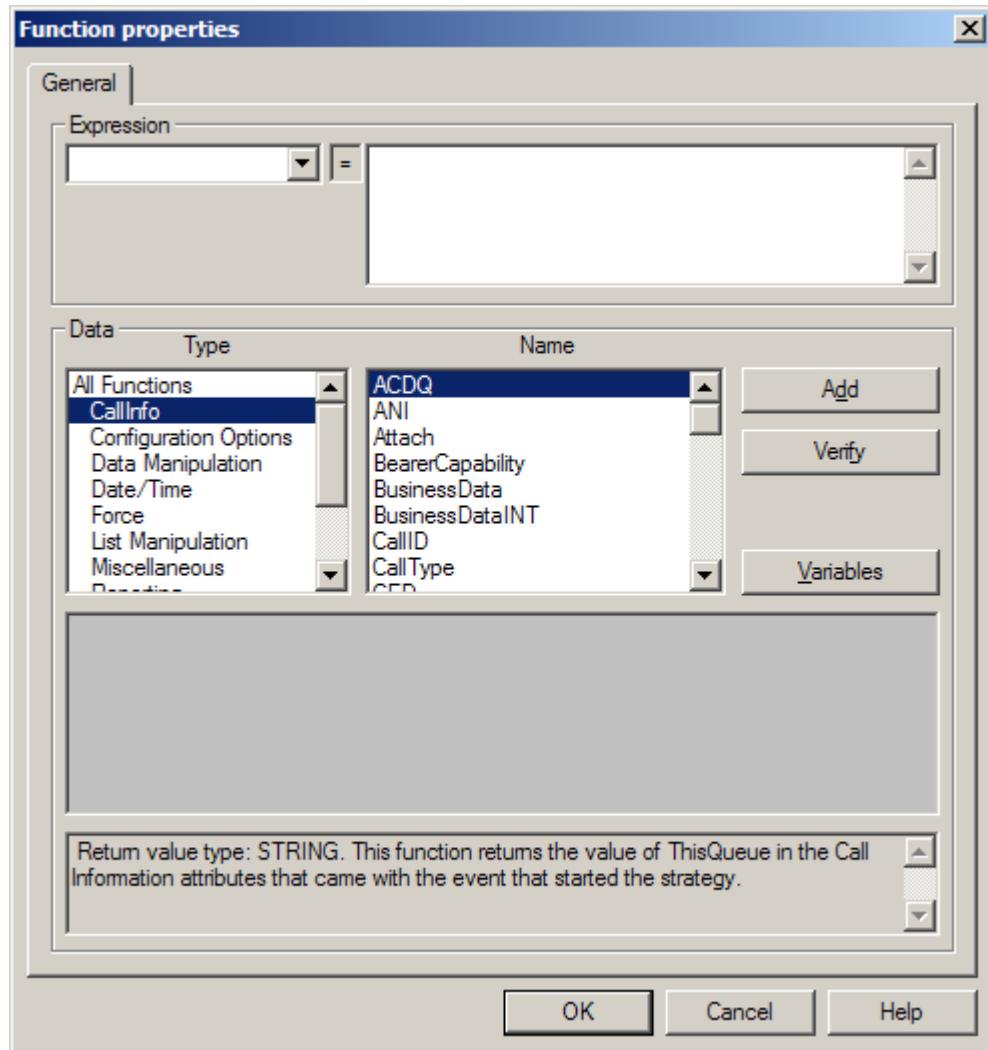


Figure 216: Function Properties Dialog Box, CallInfo Functions

Each of the functions of this class produces a specific piece of information associated with the current interaction. Accordingly, the function parameters, when they exist, specify the piece of information desired and possibly its new value. In contrast, some `Call Info` functions have no parameters at all and are shortcuts for more generic ones.

ACDQ

Parameters: none

Return value type: STRING

This function returns ThisQueue in the T-Server Event structure; ThisQueue is the directory number of the most significant ACD group with respect to the event in question. The function ACDQ is generally used in special circumstances in which customers need to consider a ThisQueue message from T-Server.

ANI

Parameters: none

Return value type: STRING

Automatic Number Identification; sent as an attribute of the event that started the strategy.

Attach

Warning! Because e-mail routing strategies used in Multi-Channel Routing interaction workflows can recycle e-mails through several routing strategies, do not use the Attach function in e-mail routing strategies. Use the Update function instead (see [page 467](#)).

Parameters: Key: STRING (Interaction Data, constant or variable)

Value: STRING (constant or variable)

Return value type: VOID

Use to attach data, such as information entered by the caller, to an interaction. The Attach function takes the retrieved information and instructs URS to provide this information to T-Server, ultimately making it available to the target agent.

This feature attaches a new key-value pair to the existing interaction User Data structure (or creates the structure if it does not exist). Old interaction User Data (if any) with the same key are not removed. Key and Value are both terms that describe how information is stored in the T-Server interaction User Data structure. Key designates the category or categories by which information is organized.

Keys can be specified in two ways:

1. Enter the key manually.

2. Opposite Key, click in the Value field. In the resulting Key dialog box, Select a Type (Interaction Data or Variable), and then select or enter a value for the key. The values available for Interaction Data Type are the Interaction Data included in the Interaction Data list. The values available for the Variable type are the variables included in the variable list.

Values can be specified in the following ways:

1. Enter the value manually if the key was entered manually.
2. Select a variable from the list of variables if the key was selected as variable.
3. Select a value from the list of selected Interaction Data values or variables if the key was selected as Interaction data.

Important Information

- Every Attach function results one attachment request to T-Server.
- More than one key-value pair can be attached with one Attach function. If key is empty, then URS will get both keys and values from the value parameter. The value parameter is considered in such cases as a complex string in the STRING (list) format: key1:value1| key2:value2| .
- The MultiAttach (see [page 140](#)) object can be used to combine more than one Attach (Update) function into one attachment request to T-Server. Genesys recommends to use the Multi Attach object if you need to use multiple Attach or Update functions (from a performance and strategy size point of view).
- Functions Attach and Update are *set* functions; they modify interaction User Data. Functions UData and UdataINT are complimentary *get* functions, they read interaction User Data.
- Simple key values are attached to the interaction in string format. If, however, the key has the form Key1.Key2.....KeyN-1.KeyN, then a sequence of nested sublists is created. The interaction User Data will contain a sublist with key Key1, which itself will contain a sublist with Key2, and so on. The innermost one will contain a key-value pair with KeyN and a specified string value.

Warning! If you use the Update or Attach function and specify multiple key-value pairs, URS removes all commas, spaces, and new line characters that you place at the end of the values you enter. As a result, any value consisting entirely of these characters is reduced to an empty string.

Size Limitation For Key-Value Strings

In key-value strings (example: 'key1:value1|key2:value2|....')

- The distance between two adjacent pipes ('|'), or from the beginning of the string to first pipe, or from the last pipe to end of string, should not be more than 4,000 bytes.
- The size of one key-value pair must be less than 4,000 bytes.
- The distance from pipe to colon (':') should not be more than 1,000 bytes.
- The size of one key should not be more than 1,000 bytes.

BearerCapability

Parameters: none

Return value type: STRING

This function retrieves from the Extensions list of the current interaction the value corresponding to the key BEARER_CAP. Calling this function is equivalent to calling `ExtensionData['BEARER_CAP']`.

For a limited number of Network T-Servers, an interaction Extensions list will already contain a key-value pair with the key BEARER_CAP at the moment the interaction arrives. The corresponding value will be a number designating the type of device from which the interaction originated (telephone, modem, fax, and so on). Consult the Network T-Server Reference Manual for your specific Network T-Server to determine whether this feature is supported and on the meaning of the different values.

BusinessData

Parameters: Key: STRING (Business Data, constant or variable)

Return value type: STRING

This function retrieves the value of the specified from the User Data information collected from the incoming interaction when the customer responds to the IVR or an agent responds. The User Data information is stored throughout the lifetime of the interaction in the form of key-value pairs. You must make sure that an appropriate pair is attached earlier either by the caller (in the case of a voice routing strategy) or by an agent or via the strategy itself. See Figure 49 on [page 103](#) for a use case.

The difference between the functions BusinessData and InteractionData is the *source* of data:

- For BusinessData, IRD provides a list of Business Attributes (see Figure 48 on [page 102](#)).

- For `InteractionData`, IRD provides list of Interaction Data defined in IRD (see Figure 25 on [page 72](#)), which are transaction objects of type `InteractionData` in Configuration Manager. This function requires a 7.0.1 or later Framework environment.

Note: Configuration Layer supports a `Business Attribute` that represent a `Global Interaction Type` (values are `Outbound`, `Inbound`, and `Internal`). You can use this attribute to track the original purpose of an interaction that has moved across T-Servers (Media Servers). You can use the standard user data access functions (`UData[key]`, `InteractionData[key]`, `BusinessData[key]`) with any `Business Attribute` including this one.

BusinessDataINT

Parameters: Key: STRING (Business Data, constant or variable)

Return value type: INTEGER

This function retrieves the value of the specified key from User Data information collected from the incoming interaction when the customer responds to an IVR or an agent responds. The User Data information is stored throughout the lifetime of the interaction in the form of key-value pairs. You must make sure that an appropriate pair is attached earlier either by the caller (in the case of a voice routing strategy) or by an agent or via the strategy itself.

The difference between the functions `BusinessDataINT` and `InteractionDataINT` is the *source* of data:

- For `BusinessDataINT`, IRD provides a list of `Business Attributes` (see Figure 49 on [page 103](#)).

For `InteractionDataINT`, IRD provides list of Interaction Data defined in IRD (see Figure 25 on [page 72](#)), which are transaction objects of type `InteractionData` in Configuration Manager. This function requires a Framework 7.0.1 or later environment.

CallID

Parameters: none

Return value type: INTEGER

The `CallID` is the identification number given to each interaction by the switch. This function can enable Custom Server to create routing scenarios or track interactions based on either of these IDs.

CallType

Parameters: none

Return value type: INTEGER

This function returns a number corresponding to the type of the current interaction. The returned value can be one of the following:

- 0·CallTypeUnknown
- 1·CallTypeInternal
- 2·CallTypeInbound
- 3·CallTypeOutbound
- 4·CallTypeConsult

CallUUID

Parameters: none

Return value type: STRING

Use to access T-Library attribute CallUUID. Once the strategy returns CallUUID, you can then attach the value to the interaction User Data or print it in a log file for the purpose of testing and verifying new strategy call flows.

Note: If this function is not available for selection in IRD's Function properties dialog box, you must manually enter `CallUUID[]` in the Expression pane of the dialog box on the right side of the equal sign. This workaround is only available starting from IRD version 7.6.100.07 or later.

CED

Parameters: none

Return value type: STRING

This function returns the digits collected from the caller and attached to the interaction.

In 7.6, the way URS updates a call's attribute CED (Collected Digits) is changed. URS now updates CED as follows:

- Upon receiving any event from the following list (if event has not empty `AttributeCollectedDigits`):
`EventQueued, EventRouteRequest, EventRinging, EventEstablished, EventPartyChanged`
- Upon receiving `EventDigitsCollected`
- Upon receiving `EventTreatmentEnd` for currently played treatment.

ConnID

Parameters: none

Return value type: STRING

The ConnID (or Connection ID) is a number T-Server assigns to identify and track each interaction. This function can enable Custom Server to create routing scenarios or track interactions based on this ID.

DeleteAttachedData

Parameter: Key: STRING (constant or variable)

Return value type: VOID

URS can access interaction User Data in T-Server Event messages (EventRouteRequest in most cases). Use DeleteAttachedData[key] to delete interaction User Data (attached data). If an asterisk (“*”) is used as a key value, the entire attached data is deleted. For more information on using, see “Working with User Data” on [page 303](#).

Dest

Parameters: none

Return value type: STRING

This function returns the value in the ThisDN field in the T-Server Event structure.

DNIS

Parameters: none

Return value type: STRING

Dialed Number Identification Service; sent as an attribute of the event that started the strategy.

ExtensionAttach

Parameters: Key: STRING (constant or variable)

Value: STRING (constant or variable)

Return value type: VOID

This function adds a key-value pair specified by an argument in the Extensions list of the current interaction. It is the only way this function differs from function Attach. Extensions (as opposed to User Data) are short-lived interaction data. They exist only while interaction is routed and are not transferred to the agent.

Functions ExtensionAttach and ExtensionUpdate are *set* functions: they modify interaction extensions. Function ExtensionData is complimentary *get* function; it reads interaction User Data.

Simple key values are attached to the interaction in string format. If, however, a key has the form Key1.Key2...KeyN-1.KeyN, then a sequence of nested

sublists is created. An interaction extension will contain sublists: Key1 will contain a sublist with Key2, and so on. The innermost sublist will contain a key-value pair with KeyN and specified string value.

ExtensionData

Parameter: Key: STRING (constant or variable)

Return value type: STRING (constant or variable)

This function returns the first value associated with a specified key from the Extensions list of the current interaction.

Function ExtensionData is get function: it reads interaction Extensions. Function ExtensionAttach and ExtensionUpdate are complimentary set functions: they modify interaction Extensions.

If Key has a form Key1.Key2...KeyN-1.KeyN, then Extensions will be searched for in the sublist with Key1. Key1 itself will be searched for a sublist with key Key2 and so on. The inner most sublist will be searched for the element with KeyN, which will be returned.

This function can be used to retrieve values of Extensions attribute for a scenario where a failure to apply treatment occurs (EventTreatmentNotApplied).

ExtensionUpdate

Parameters: Key: STRING (constant or variable)

Value: STRING (constant or variable)

The ExtensionUpdate function behaves exactly like ExtensionAttach (see [page 456](#)). The only difference is that ExtensionUpdate removes old interaction Extensions with the specified Key before attaching new ones.

FirstHomeLocation

Parameters: none

Return value type: STRING

This function returns the value in the FirstTransferHomeLocation field in the T-Server event that started the strategy. Usually such event is EventRouteRequest.

GetCurrentLeg

Parameters: none

Return value type: STRING(LIST)

This function returns information about the DN where the interaction is located at the time the function is called. The information is provided in the form of a string list with the following attributes:

- **type**—The numeric value of the DN type (Extension = 1, Routing Point = 4), according to the Configuration library.
- **number**—The DN number.
- **cdn**—The value is 1 if it is Queue type DN (RouteRequest, RouteUsed events are distributed), and 0 if it is not (Ringing, Established, Released events are distributed).
- **event**—The numeric presentation of the last event received for this DN for the current call (according to the TLibrary) - 71(EventRouteRequest), 60(EventRinging), 64(EventEstablished).

If the interaction has no DN, the function returns an empty string.

Interaction Routing Designer has named numeric constants that present types of events and DNs. They can be used along with raw numeric values.

Examples include the following:

`CFGExtension`, `CFGACDPosition`, `CFGRoutingPoint`, and
`EventRinging`, `EventEstablished`

GetCurrentSwitch

Parameters: none

Return value type: STRING

This function returns the configured name of the switch on which the interaction is located at the time when the function is called. This switch is determined on the basis of the last relevant event that was received from T-Server.

GetCurrentTServer

Note: Use the ActiveServerName function (see [page 505](#)) to return the name of the active T-Server from the pair working in Hot Standby or Warm Standby mode.

Parameters: none

Return value type: STRING

This function returns the name of the T-Server Application for the current interaction. For both primary and backup T-Servers, the T-Server name entered in the Connections tab of the URS Application is returned as the current T-Server.

Note: This function is not designed to differentiate between primary and backup T-Servers. For both, the name of the T-Server that URS considers configured as primary is returned. URS considers a T-Server as primary if it is in URS's Connections list.

GetCustomerSegment

Parameters: none

Return value type: STRING

Gets the requested Customer Segment contained in the interaction. In this release, CustomerSegment is implemented as User Data so this function is a shortcut for UData[CustomerSegment].

For additional information on Customer Segment, see the MultiAttach object on [page 140](#). For information on UData, see [page 465](#).

GetMediaType

Note: Because it is limited to the predefined media types associated with integers (see Table 131 on [page 459](#)), the GetMediaType function has limited applicability in IRD 7.0.1 or later. Genesys recommends using GetMediaTypeName (see [page 474](#)) instead, which supports both predefined and custom media type names. Medias that do not have any predefined integers cannot be handled with the GetMediaType function.

Parameters: none

Return value type: INTEGER

This function returns the interaction media type assigned by T-Server. It is a read-only property of the interaction and cannot be modified in a strategy.

In IRD, this function can be used in IF statements and Generic Segmentation objects to compare it to the constants shown in the first column of [Table 131](#).

Table 131: Media Types Integers

Constant and Integer	Supported in IRD 7.0	Supported in IRD 7.0.1 and later
TMediaVoice-voice (PSTN) call; returned integer=0	yes	yes
TMediaVoIP-voice over IP; returned integer=1	yes	yes
TMediaEMail-Electronic mail; returned integer=2	yes	yes

Table 131: Media Types Integers (Continued)

Constant and Integer	Supported in IRD 7.0	Supported in IRD 7.0.1 and later
TMediaVMail-voice mail; returned integer=3	yes	yes
TMediaSMail-scanned mail; returned integer=4	yes	yes
TMediaChat-chat session; returned integer=5	yes	yes
TMediaVideo-video; returned integer=6	yes	yes
TMediaCobrowsing-cobrowsing; returned integer=7	yes	yes
TMediaWhiteboard-whiteboard; returned integer=8	yes	yes
TMediaAppSharing-application sharing; returned integer=9	yes	yes
TMediaWebForm-webform; returned integer=10	yes	yes
TMediaWorkItem-work item (generic); returned integer=11	yes	yes

GetRoutingPoint

Parameters: none

Return value type: STRING

This function returns the configured name of the DN from which the current interaction is being routed.

GetServiceObjective

Parameters: none

Return value type: STRING

Gets the Service Objective (see Figure 80 on [page 145](#)) contained in the interaction. Because Service Objective is currently implemented as User Data, this function is a shortcut for function UDataINT(ServiceObjective) described on [page 554](#).

- See “MultiAttach” on [page 140](#) for more information on Service Objective.
- See the PriorityTuning function on [page 603](#) for information on how Service Objectives can be used to prioritize interactions.

GetServiceType

Parameters: none

Return value type: STRING

Gets the requested Service Type contained in the interaction. Because Service Type is currently implemented as User Data, this is a shortcut for function UData(ServiceType).

InformationDigits

Parameters: none

Return value type: STRING

This function retrieves from the current interaction's Extensions list the value corresponding to the key INFO_DIGITS. Calling this function is equivalent to calling ExtensionData['INFO_DIGITS'].

For a limited number of Network T-Servers, an interaction's Extensions list will already contain a key-value pair with the key INFO_DIGITS at the moment the interaction arrives. The corresponding value will be a two-digit code for certain information on the origination point of the interaction. For instance, it might report that the interaction comes from a pay phone or that it comes from a telephone requiring special operator handling. This information can prove useful for routing decisions or for certain types of call blocking.

Consult the appropriate T-Server guide on whether this feature is supported and on the meaning of the different values.

InteractionData

Note: The difference between the functions InteractionData and BusinessData is the *source* of data. For BusinessData, IRD provides a list of Business Attributes (see Figure 49 on page 103). For InteractionData, IRD provides list of Interaction Data defined in IRD (see Figure 25 on page 72), which are transaction objects of type InteractionData in Configuration Manager.

Parameters: none

Return value type: STRING

This function retrieves the User Data information collected from an incoming interaction when the customer responds to either an IVR prompt (in the case of a voice routing strategy) or to an agent. The User Data retrieved is stored throughout the lifetime of the interaction in the form of key-value pairs.

Note: You must make sure that an appropriate pair is attached earlier either by the caller or the agent or in the strategy itself.

See Note below.

InteractionDataINT

Parameters: none

Return value type: INTEGER

These functions retrieve the User Data information collected from an incoming interaction when the customer responds to either an IVR prompt (in the case of a voice routing strategy) or to an agent. The User Data retrieved is stored throughout the lifetime of the interaction in the form of key-value pairs. Use InteractionDataINT with the Windows operating system.

You must make sure that an appropriate pair is attached earlier either by the caller or the agent or in the strategy itself.

Note: Configuration Layer supports a `Business Attribute` that represent a `Global Interaction Type` (values are `Outbound`, `Inbound`, and `Internal`). You can use this attribute to track the original purpose of an interaction that has moved across T-Servers (Media Servers). You can use the standard user data access functions (`UData[key]`, `InteractionData[key]`, `BusinessData[key]`) with any `Business Attribute` including this one.

LATA

Parameters: none

Return value type: STRING

This function retrieves from the Extensions list of the current interaction the value corresponding to the key LATA. Calling this function is equivalent to calling `ExtensionData['LATA']`.

For a limited number of Network T-Servers, an interaction's Extensions list will already contain a key-value pair with the key LATA at the moment the interaction arrives. The corresponding value will designate the LATA from which the interaction originated. Consult the appropriate T-Server guide on whether this feature is supported.

Note: A LATA (Local Access Transport Area) is a geographical area in which the local telephone company is allowed to carry local calls and also long distance toll calls. The geographical areas were defined after the breakup of the AT&T monopoly in 1984.

NPA

Parameters: none

Return value type: STRING

This function checks whether the ANI is long enough to contain an area code and, if so, it returns the first three digits of the ANI of the current interaction. It provides a convenient way to retrieve the area code of a caller in the United States and to make routing decisions based on it.

NPANXX

Parameters: none

Return value type: STRING

This function checks whether the ANI is long enough to contain an area code and, if so, it returns the first six digits of the ANI of the current interaction. It provides a convenient way to retrieve the area code and the prefix of a caller in the United States.

Orig

Parameters: none

Return value type: STRING

This function returns the OtherDN field in the T-Server Event structure.

OtherTrunk

Parameters: none

Return value type: INTEGER

This function returns the value from AttributeOtherTrunk which is taken from the event that starts the strategy.

PACCode

Parameters: none

Return value type: INTEGER

This function is equivalent to the CED function described on [page 455](#).

PACType

Parameters: none

Return value type: INTEGER

This function retrieves from the Extensions list of the current interaction the value corresponding to the key PAC_TYPE. Calling this function is equivalent to calling ExtensionData['PAC_TYPE'].

For a limited number of Network T-Servers, an interaction's Extensions list will already contain a key-value pair with the key PAC_TYPE at the moment the

interaction arrives. Consult the appropriate T-Server guide on whether this feature is supported and on the meaning of the corresponding value.

RequestType

Parameters: none

Return value type: INTEGER

This function retrieves from the Extensions list of the current interaction the value corresponding to the key REQUEST_TYPE. Calling this function is equivalent to calling `ExtensionData['REQUEST_TYPE']`.

For a limited number of Network T-Servers, an interaction's Extensions list will already contain a key-value pair with the key REQUEST_TYPE at the moment the interaction arrives. Consult the appropriate T-Server guide on whether this feature is supported and on the meaning of the corresponding value.

SetHomeLocation

Parameters: HomeLocation (format: tserver@switch)

Example: `SetHomeLocation['Env_TServer1@Env_Switch']`

Return value type: VOID

The function allows you to specify the home location for a call, which is the switch `SetHomeLocation` and T-Server from which the call originated. In cases in which URS needs this information, it can be supplied using this function.

For every call, at the very beginning URS will execute code equivalent to `SetHomeLocation[FirstHomeLocation[]]`. This means that `FirstTransferHomeLocation` will be used by default as the call home location (see function `FirstHomeLocation` on [page 457](#)).

You can use this function to implement the prioritization mechanism described below to select T-Server for Virtual Queues events distribution.

Distributing Virtual Queue Events

URS uses a prioritization mechanism to select T-Servers for distributing virtual queue events. Starting with 7.6, URS takes into account the very first T-Server that the call was transferred from (if any). To preserve backward compatibility, when selecting T-Servers for distributing the next virtual queue event, the current order that URS uses is as follows:

1. The T-Server already used to send events for this virtual queue for the current call. Applicable only if the current virtual queue event is not the first one distributed for the current call.
2. The T-Server on which strategy execution was started.

3. The T-Server with the name taken from `AttributeFirstTransferHomeLocation` of the `EventRouteRequest` (or other requests that start strategy execution).

4. All other T-Servers that URS is connected to in arbitrary order.

Based on the above priority, URS uses the first T-Server found that it is connected to and that monitors the virtual queue. If no such T-Server is found, no virtual queue event is distributed.

StateCode

Parameters: none

Return value type: STRING

This function retrieves from the Extensions list of the current interaction the value corresponding to the key `US_STATE`. Calling this function is equivalent to calling `ExtensionData['US_STATE']`.

For a limited number of Network T-Servers, an interaction's Extensions list will already contain a key-value pair with the key `US_STATE` at the moment the interaction arrives. Consult the appropriate T-Server guide on whether this feature is supported and on the meaning of the corresponding value.

ThisTrunk

Parameters: none

Return value type: INTEGER

This function returns the value from `AttributeThisTrunk` which is taken from the event that starts the strategy.

UData

Parameter: Key : STRING (Interaction Data, constant, or variable)

Return value type: STRING

This function retrieves the User Data information associated with the interaction, such as when the caller responds to an IVR. This User Data information is temporarily stored on Genesys T-Server throughout the lifetime of the interaction.

- Use this function when User Data information should be considered as a string.
- Use `UDataINT` when the User Data information should be considered as an integer.

When selecting a Key, the value can be specified in two ways:

1. Enter the Key for the Interaction Data manually.

2. In the Function Properties dialog box, select All Functions (and then the function name), or select Interaction Data or Variables, and then select or enter a value. The values available for the Interaction Data type are defined Interaction Data. The values available for the Variable type are the variables included in the Variable list.

Important Information

- Functions UData and UDataINT are *get* functions: they read interaction User Data. Functions Attach and Update are complimentary *set* functions: they modify interaction User Data.
- UData always returns a string. If the userdata Key points to a sublist, it will be converted into string with list format: Key1:value1|Key2:Value2] and so on.
- Starting with release 8.0, if the UData function is used, to get the whole User Data as a string with a list format, value '*' must be used. In 7.x the using empty Key would get the same result.
If the URS option `function_compatibility` is set to a value other than 8.0, the old method (empty parameter) works as it did prior to the 8.0 release.
- If Key has the format Key1.Key2.....KeyN-1.KeyN, user data will be searched for in a sublist within Key1. Key1 will be searched for sublist with key Key2 and so on. The innermost will be searched for element with key KeyN, which will be returned.
- GetCustomerSegment (see [page 459](#)) is shortcut for function UData[CustomerSegment].
GetServiceType is shortcut for function UData[ServiceType].
GetServiceObjective is shortcut for UDataINT[ServiceObjective].
While the above functions can all be replaced with UData functions, this is not recommended. Instead, Genesys recommends using the above listed functions as shorter versions of getting specific User Data.
- Configuration Layer supports a Business Attribute that represent a Global Interaction Type (values are Outbound, Inbound, and Internal). You can use this attribute to track the original purpose of an interaction that has moved across T-Servers (Media Servers). You can use the standard user data access functions (UData[key], InteractionData[key], BusinessData[key]) with any Business Attribute including this one.

UDataINT

Parameter: Key : STRING (Interaction Data, constant or variable)

Return value type: INTEGER

This function is identical to UData function with only one exception. The UDataINT function attempts to convert the returned value to an integer.

Use UDataINT when numerical comparison is required. For example, assume that User Data for one interaction has a value of 500 and the User Data for another interaction has a value of 1000. If UData is used to get the data for the purpose of comparison, the result will be that 500 is greater (as a string) than 1000. If UDataINT is used to get the values, the result will be that 1000 is greater (as number) than 500.

If the resulting value cannot be interpreted as number, the function returns 0.

Update

Warning! Because routing strategies used in eServices business processes can recycle interactions through several routing strategies, use the Update function instead of the Attach function in eServices routing strategies.

Note: Also see “UpdateBusinessData” on [page 468](#) and “UpdateInteractionData” on [page 469](#). In a Framework 7.0.1 or later environment, Update can be used for the same purpose as the UpdateBusinessData and UpdateInteractionData. See the those function descriptions for additional information.

Parameters: Key: STRING (Interaction Data as shown in Figure 25 on [page 72](#), Business Data, constant or variable)
Value: STRING (constant or variable)

Return value type: VOID

The Update function for e-mail routing strategies is similar to the Attach function for voice routing strategies (see [page 451](#)). A difference is that the Update function removes old interaction User Data with the specified Key before attaching new data.

While the Update function requires more resources to execute than the Attach function, it has an advantage over the Attach function in that it prevents interaction User Data from growing and accumulating when the same key is used for attaching multiple times.

The Database Wizard can implicitly use the Update function if selected (see “Database Wizard” on [page 108](#)) to attach database output.

Attaching Several Key-Value Pairs

URS enables you to attach several key-value pairs in one request by using the Update function in a following way:

- Make the first argument empty.

- Make the second argument a string in the STRING (list) format, that is created from a series of key-value pairs separated by the pipe symbol (|). Every key is separated from its value by a colon (:).

The attached value is everything between the first colon and a pipe symbol.

Example

```
Update['', 'EMail_Enter_Date:11/13/2003|EMail_Enter_Time:10:32']
```

Here the EMail_Enter_Time is a key and 10:32 is a value.

See also “MultiAttach” on [page 140](#) and “Size Limitation For Key-Value Strings” on [page 453](#).

Note: See the *eServices (Multimedia) 8.1 User's Guide*, Interaction Properties chapter, for important information about Business Attributes that are user-accessible and provides information on which of them are safe to modify.

Warning! If you use the Update or Attach function and specify multiple key-value pairs, URS removes all commas, spaces, and new line characters that you place at the end of the values you enter. As a result, any value consisting entirely of these characters is reduced to an empty string.

UpdateBusinessData

Note: This function requires a Framework 7.0.1 or later environment. If connecting to a 7.0 Configuration Server database in a multi-tenant environment, IRD 7.0.1 or later will show no Business Attributes (see Figure 49 on [page 103](#)). There is no Business Attributes folder in Configuration Manager 7.0.

Parameters: Key: STRING (Business Data, constant or variable)

Value: STRING (constant or variable)

Return value type: VOID. This function updates interaction Business Attributes used in interactions, such as eServices interactions (see “The Interaction Design Window” on [page 98](#) and “Using Business Attributes” on [page 103](#)). Also see the *eServices (Multimedia) 8.1 User's Guide*, Interaction Properties chapter, for important information about Business Attributes that are user-accessible and provides information on which of them are safe to modify.

Using the UpdateBusinessData function instead of the Update function indicates what you want to update. For example, if you wish to update interaction business data, as found in Configuration Manager (see Figure 49 on

[page 103](#)), use UpdateBusinessData. In this case, IRD includes the Business Attribute in its integrity checking (see “Check Integrity Mode” on [page 34](#)). If the Business Attribute is not found during integrity checking, IRD generates a warning.

UpdateInteractionData

Parameters: Key: STRING (Interaction Data, constant or variable)
Value: STRING (constant or variable)

Return value type: VOID.

This function updates interaction user data (see “User Data” on [page 152](#)), such as user data attached to non-voice interactions. For more information on the type of data this function updates, see “Interaction Data” on [page 72](#).

Configuration Options Functions

The IRD Function Properties dialog box with Configuration Options selected is shown in [Figure 217](#).

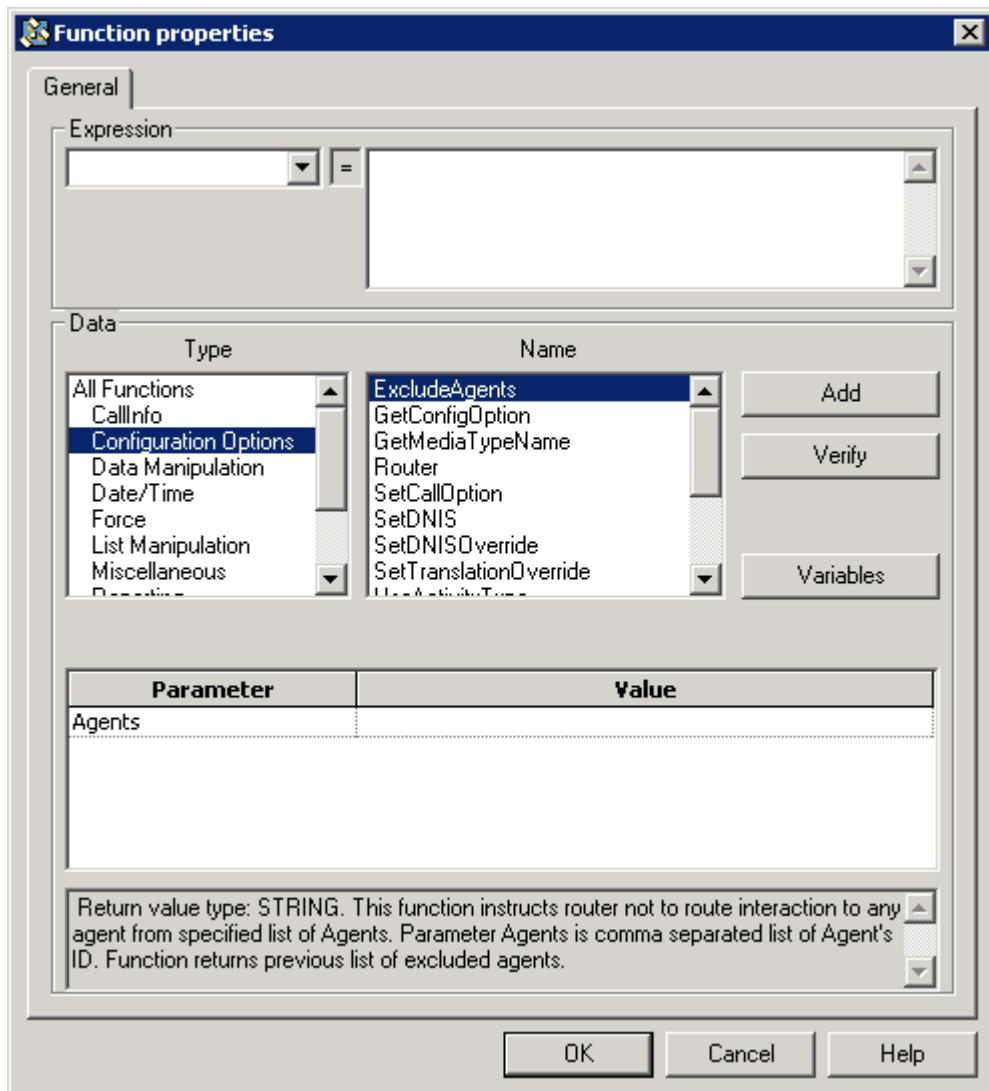


Figure 217: Function Properties Dialog Box, Configuration Option Functions

This section presents functions in the Configuration Options category (see Table 130 on [page 430](#)).

ExcludeAgents

Parameters: Agents: STRING (comma-separated list of agent IDs or variable)

Return value type: STRING

This function instructs URS not to route interactions to any agent on the specified list of agents. Parameter Agents is comma-separated list of agent IDs. Function returns the previous list of excluded agents.

Previous to 7.6, if a target was selected (if it was ready according to Stat Server and reserved) and then excluded from the list of valid targets using the

ExcludeAgents function, this target was not actually excluded. Starting with 7.6, the ExcludeAgents function does exclude the agent in the above scenario.

Warning! Function ExcludeAgents affects IVR targets.

FindConfigObject

Parameters: TYPE: INTEGER, Valid Values: CFGDN, CFGPlace

Properties: LIST or variable (provide list of search criteria)

Return value type: LIST

This function returns information about a requested configuration object. You must specify the type of object for which to search (for example, CFGPlace) and the list-presenting search criteria. A valid set of search properties consists of either the name or a combination of the switch and number.

For CFGDN objects, name specifies an alias of the required DN, switch specifies the name of the switch to which the DN must belong, and number specifies this DN number.

For CFGPlace object, name specifies the name of required place, switch specifies the name of switch to which a DN (from among the DNs belonging to this place) belongs, and number specifies the number of this DN.

This function returns an extended list of information about the object that is found. The search criteria specifies a subset of the object properties, while the function provides the rest of the data. The search criteria must be a unique subset of the properties that identify the configuration object. See the following search criteria and results:

Search Criteria FindConfigObject[CFGDN, 'name:2201_vit_sw2'] or
FindConfigObject[CFGDN, 'number:2201|switch:vit_sw2']

Results "dbid:1122|name:Place_102_vit_sw2|tenantdbid:103|tenant:Vit|#1.number:102|#1.switch:vit_sw2|#1.type:2|#2.number:112|#2.switch:vit_sw2|#2.type:1|dns:2".

All of the data about the specified object can be extracted and analyzed by using the List Manipulation function. [Table 132](#) contains a description of the fields in the returned LIST.

Table 132: Search Criteria Results—LIST Fields

Field	Description
dbid:1122	DBID of the specified place.
name:Place_102_vit_sw2	Name of the specified place.
tenantdbid:103	DBID of tenant to which the place belongs.

Table 132: Search Criteria Results—LIST Fields (Continued)

Field	Description
tenant:Vit	Name of tenant to which the place belongs.
dns:2	The number of DNs for the specified place.
dn.#1.type:2	The first DNs type. (Same type as the TargetState function.)
dn.#1.number:105	The first DNs number.
dn.#1.switch:vit_sw2	The first DNs switch.
dn.#2.type:2	The second DNs type. (Same type as the TargetState function.)
dn.#2.number:115	The second DNs number.
dn.#2.switch:vit_sw2	The second DNs switch.

Note: When searching in configuration URS, ensure that the name of switch and DN number that identifies the searched object (DN or Place) is unique. Otherwise, URS could return any one of the matching config objects.

If an object is not found by using the search criteria, the function returns an empty list and raises an `0018 Unknown object` error code.

GetConfigOption

Parameters: Option: STRING or variable (representing a string for the option)
 Lookup Sequence: `StartFromCDN`, `StartFromRouter`, `StartFromTenant`, `StartFromTserver`, or variable (representing the lookup sequence)

Return value type: STRING

This function allows the use of customized configuration options—the user can configure any option name that is different from the standard options and then use its value in routing. In particular, any options can be specified in addition to the required ones and given meaning in the strategy.

This function retrieves the current value of any URS configuration option for use in a strategy. The search for the option starts with the object properties given by Lookup Sequence (the DN from where the interaction is routed, the T-Server application controlling this DN, the tenant to which they belong, or the URS application). If the option is not found there, the search continues in the object properties corresponding to greater values of Lookup Sequence, in

increasing order, until the option is found. The empty string is returned if the option is not found.

Note: The list reusable object and the ability to define a key-value pair for a list element is an extension or variation of the GetConfigOption function (see [page 472](#)).

Starting in URS 7.5, if the option name (first parameter) is an empty string and the lookup sequence (second parameter) is specified, the function returns the object name in the context of the lookup sequence as follows:

```
Route Point name for StartFromCDN
URS name for StartFromRouter
Tenant name for StartFromTenant
T-Server name for StartFromTserver
```

Note: In URS releases prior to 7.5, if the option name was entered as an empty string, the look up sequence returned an empty string.

GetCurrentScript

Parameters: None

Return value type: STRING

This function returns the currently executed strategy name.

GetObjectProperty

Parameters:

Type: INTEGER - type of the configuration object from which the option will be obtained.

Subtype: INTEGER - type of Transaction object; required if Type parameter is Transaction. Use the CfgTransactionType number equivalents.

Name: STRING - name of the configuration object from which the option will be obtained. Name for the Person object is employeeID and for the DN object it is Alias.

Section: STRING - name of the section on the Annex or Options tab from which the option value will be obtained.

Item: STRING - name of the option.

Return value type: STRING

This function returns the value of the requested option of the configuration object identified by Type, Subtype, and Name. The value is retrieved from the Annex or Options tab, if present.

- For the list of `CfgTransactions` enumerations, refer to the [Genesys PSDK 8.1.4 Configuration Layer Enumeration List](#).

GetMediaTypeName

Note: This function (which requires Framework 7.0.1 or later), along with the `UseMediaType` function (see [page 479](#)), the `StatAgentLoadingMedia` statistic (see [page 724](#)), and the `use_agent_capacity` option ([page 674](#)) all support the Genesys agent capacity distribution model as described in the *Universal Routing 8.1 Deployment Guide*.

Parameters: `Media` (the current interaction media type, which includes all media types defined for tenant)

Return value type: STRING

Returns the name of specified `Media` as defined in the Configuration Manager, in the `Business Attributes > Media Types` folder (see “The Interaction Design Window” on [page 98](#)). The `GetMediaTypeName` function makes it for a strategy to analyze custom media types.

For example, assume you define a custom media type called `MyMedia` in Configuration Manager. A strategy cannot handle this media type using the `GetMediaType` function. Previously in IRD 6.x/7.0 using the `GetMediaType` function (see [page 459](#)):

```
GetMediaType[] = TMediaEMail
```

The `GetMediaType` function can be used only for Genesys provided-media types that have an associated integer value as (see Table 131 on [page 459](#)).

Using the equivalent (and more generic) IRD 7.0.1 and later `GetMediaTypeName` function:

```
media = GetMediaType[]
GetMediaTypeName[media] = 'Email'
```

Using `GetMediaTypeName` to get the media type enables strategies to work with any customer-defined media type.

Router

Parameters: `Get: INTEGER` (describing requested information).
`APPLICATION_NAME` or `CLUSTER_NAME` can be selected.

Return value type: STRING (requested information)

If `APPLICATION_NAME` is requested, this function returns the name of the URS that is running. If `CLUSTER_NAME` is requested, this function returns the name of the primary URS based on configuration information.

SetCallOption

Parameters: Option: STRING (variable) or one of the following constants:
`default_destination`, `request_timeout`, `null_value`,
`default_object`, `use_ivr_info`, `use_agentid`
 New option value: STRING (constant or variable)

Return value type: VOID

This function is an interaction level override for server level options. It enables the strategy to control certain options instead of URS. The options that can be overridden are:

- `request_timeout`
- `null_value`
- `default_object`
- `use_ivr_info`
- `default_destination`
- `use_agentid`
- `use_extrouter`
- `use_extrouting_type`

Note: Using this function may negatively impact URS performance.

SetDNIS

Parameter: DNIS: STRING or variable (representing a string for the DNIS)

Return value type: VOID

This function replaces the DNIS of the interaction by the value of the argument. If a call to `SetDNIS` is encountered, any DNIS override specified by another function, such as `SetDNSOverride`, will be disregarded, whether the other function is called before or after `SetDNIS`.

Important Information

- Whenever URS changes the DNIS of an interaction, use the routing type `RouteTypeOverwriteDNIS` to make the new DNIS known to T-Server.

SetDNSOverride

Parameters: Parameter: `UseNone`, `UseANI`, `UseDNIS`, `UseConfig`, or variable

Return value type: VOID

This function instructs URS how to overwrite the DNIS of the current interaction.

Important Information

- Whenever URS changes the DNIS of an interaction, use the routing type `RouteTypeOverwriteDNIS` to make the new DNIS known to T-Server.

SetObjectProperty

Parameters:

Type: INTEGER - type of the configuration object for which the Annexes are changed.

Subtype: INTEGER - type of Transaction object; required if Type parameter is Transaction. Use the `CfgTransactionType` number equivalents.

Name: STRING - name of the configuration object for which the Annexes are changed. Name for the Person object is `EmployeeID` and for the DN object it is `Alias`.

Section: STRING - name of the section in the Annex of the configuration object in which the change will be made.

Item: STRING - name of the option and its value, which are specified in pairs. The first element of the pair is the name of the option and the second element is its value.

Return value type: VOID

This function sets option names and values on the Annex tab of the specified object identified by Type, Subtype, and Name.

For this function to run successfully, the URS application must be running under a Configuration Server account that allows modifications of the configuration object.

- For the list of `CfgTransactions` enumerations, refer to the [Genesys PSDK 8.1.4 Configuration Layer Enumeration List](#).

SetTranslationOverride

Parameters: Source DN: STRING or variable (representing a string for the Source DN)

Destination DN: STRING or variable (representing a string for the Destination DN)

Location: STRING or variable (representing a string for the location)

Route Type: `RouteTypeUnknown`, `RouteTypeDefault`, `RouteTypeLabel`, `RouteTypeOverwriteDNIS`, `RouteTypeDDD`, `RouteTypeIDDD`, `RouteTypeDirect`, `RouteTypeReject`, `RouteTypeAnnouncement`, `RouteTypePostFeature`,

`RouteTypeDirectAgent`, `RouteTypePriority`,
`RouteTypeDirectPriority`, or variable

DNIS: STRING or variable (representing a string for the DNIS)

Reason: STRING (list) or variable (representing a string for the reason)

Extension: STRING (list) or variable (representing a string for the extension)

Return value type: VOID

This function overrides any translation specified in configured Switch Access Codes for routing to remote targets later in the strategy, and instructs URS to use the information specified in the parameters for the purpose of number translation. For additional information, see “Number Translation” on [page 705](#).

Note: In Interaction Routing Designer, preexisting DNs are not shown in the drop-down lists under the parameters SourceDN and DestinationDN.

UseActivityType

Parameters: Activity: any, default

Return value type: VOID

This function affects an agent’s *activity*; a concept that URS/IRD currently use internally to implement routing to Campaign groups and integration with OCS.

If the activity of the agent (as assigned by OCS) is the same as what’s in the activity queue, the call will wait for this agent (the call can be routed to this agent). If the OCS-assigned activity is not the same as what’s in the activity queue, the call will not be routed to the agent.

Two predefined activities (`default` and `any`) are to be used with this function.

After invoking `UseActivityType` with parameter `any`, every queue where the call is placed ignores activity checking, and such calls can be delivered to any agent no matter what his/her assignment may be.

After invoking `UseActivityType` with parameter `default`, every queue where the call is placed will be checked for activity matching, and URS will behave as it does now; delivering calls to agents with a matching assignment.

The `UseActivityType` function affects only placement of a call in a queue *after* it is invoked. Just invoking `UseActivityType['any']` will not route an inbound call to an agent assigned to some outbound campaign group; the call should enter after that into one or more target selection objects. As a result, it is possible to make inbound calls wait simultaneously for some set of agents for routing only if they are assigned to Inbound, and another set of agents for routing no matter what their assignment.

UseCustomAgentType

Parameters: Property: STRING (variable or constant)

Value: STRING (variable or constant)

Return value type: VOID

This function instructs URS on the Value of an Agent property that an agent must have to be considered a valid target for the current interaction. The property is specified inside the folder with the name of the URS Application, in the Annex tab of the Person object.

URS checks the agent type for the following targets: Agents, Agent Groups, Places, Place Groups. For last two targets, the agent type can be verified only if Stat Server reports the name of the agent associated with the Place in question.

UseCustomDNTYPE

Parameters: Property: STRING (variable or constant)

Value: STRING (variable or constant)

Return value type: VOID

This function instructs URS on the Value of a DN Property that a DN must have to be considered a valid target for current interaction. The property is specified under Switches, inside the folder with the name of the URS Application, in the Annex tab of the DN.

This function is similar to the function UseDNTYPE. The only difference is that the UseCustomDNTYPE function allows you to define your own types of DNs (create custom DN types).

URS checks the DN type for the following targets: Agents, Agent Groups, Places, Place Groups, Routing Points, Queues.

UseCustomPlaceType

Parameters: Property: STRING (variable or constant)

Value: STRING (variable or constant)

Return value type: VOID

This function instructs URS on the Value of a Place property that a Place must have to be considered a valid target for current interaction. The property is specified in the Annex of Place inside the folder with the name of URS Application.

URS checks the Place type for following targets: Agents, Agent Groups, Places, Place Groups.

UseDNTType

Parameters: Type of DN: any, CFGACDPosition, CFGCellular, CFGChat, CFGCoBrowse, CFGEmail, CFGExtension, CFGFax, CFGVideo, CFGVoicemail, CFGVoIP, CFGWorkflow, or variable

Return value type: VOID

This function instructs URS on the type of DN to use when routing to a target. Choose the DN based type of interaction being routed.

This delivers various interactions to the correct DN on an agent's desktop.

Important Information

- URS sets the default DN type from the MediaType attribute of TEvent. UseDNTType function overrides the default.
- See UseMediaType function.

UseMediaType

Note: The UseMediaType function, the GetMediaTypeName function (see [page 474](#)), the StatAgentLoadingMedia statistic (see [page 724](#)), and the use_agent_capacity option ([page 674](#)) all support the Genesys agent capacity distribution model.

Parameters: Media (all Media Types defined for tenant)

Return value type: VOID

This function instructs URS on the media types (examples: voice, e-mail, chat) that a target is associated with. In order to accept interactions of a particular media type, a target must be associated with that media type.

URS sets the initial media type from the Media Type attribute of TEvent if the interaction is not a voice interaction. The UseMediaType function overrides this initial setting.

URS can pick up for routing either the available media of an agent or a ready DN of an agent.

Important Information

- For backward compatibility if some DN type (DNTYPE) has a corresponding media (MEDIA) associated (for example e-mail), then UseMediaType[MEDIA] is equivalent to UseDNTType[DNTYPE].
- UseMediaType[Voice] indicates URS will select only Extensions or ACDPositions of agent.

Data Manipulation Functions

The IRD Function Properties dialog box with Data Manipulation selected is shown in [Figure 218](#).

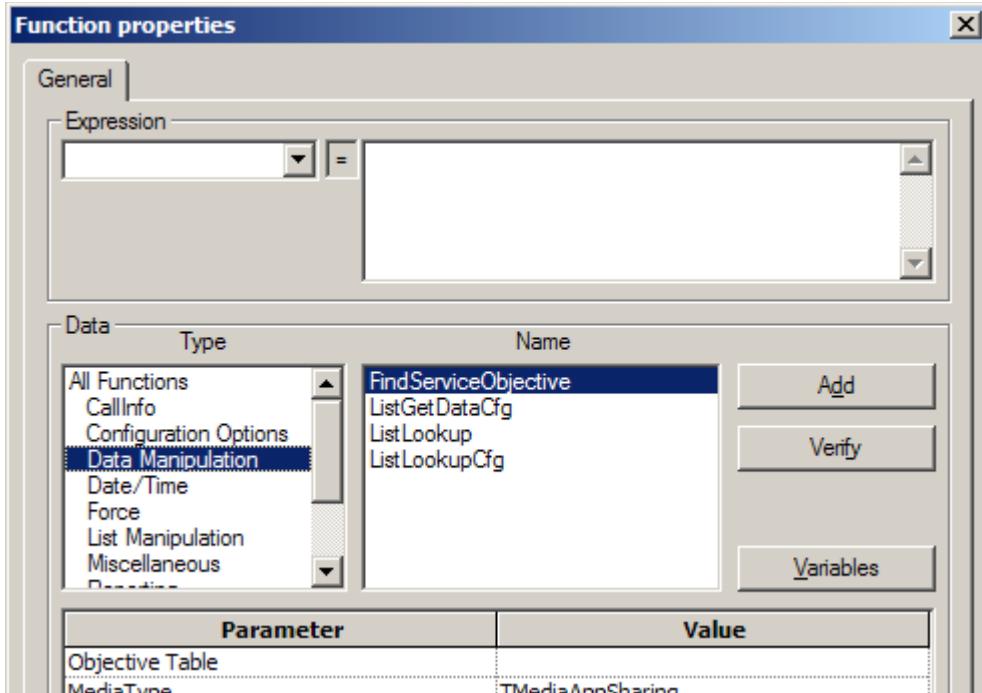


Figure 218: Function Properties Dialog Box, Data Manipulation Functions

The last three functions in Figure 218 on [page 480](#) make it possible to read and write values in the Extensions key-value list when the strategy supplies explicit instruction. The Extensions list can be a nested key-value list; that is, the value for a given key can be a key-value list itself, which can also be nested.

FindServiceObjective

Note: For additional information, see the section on business-priority routing in the *Universal Routing 8.0 Routing Application Configuration Guide*, Recommended Settings table. Also, this function requires Framework 7.0.1 or later; it cannot be used with a Framework 7.0 environment. If connecting to a 7.0 Configuration Server database in a multi-tenant environment, IRD shows no Business Attributes (see Figure 49 on [page 103](#)).

Parameters: ObjectiveTable: STRING (Objective Table Name, constant or variable)

MediaType: INTEGER (MediaType, constant or variable)

ServiceType: STRING (ServiceType, constant or variable)

`Customer Segment: STRING (CustomerSegment, constant or variable)`

`Update: INTEGER (constant or variable)`

Return value type: `INTEGER`

Finds the Service Objective for a given combination of Customer Segment, Service Type, and Media Type.

The returned value is predefined in a Configuration Layer Objective table that defines a Service Objective for various combinations of Customer Segment, Service Type, and Media Type (see Figure 80 on [page 145](#)).

If the `Update` parameter is true (see Figure 218 on [page 480](#)), the Service Objective with Service Type and Customer Segment are automatically attached to the interaction. In this case, no special strategy step is required to associate the Customer Segment, ServiceType, Service Objective triplet with the interaction.

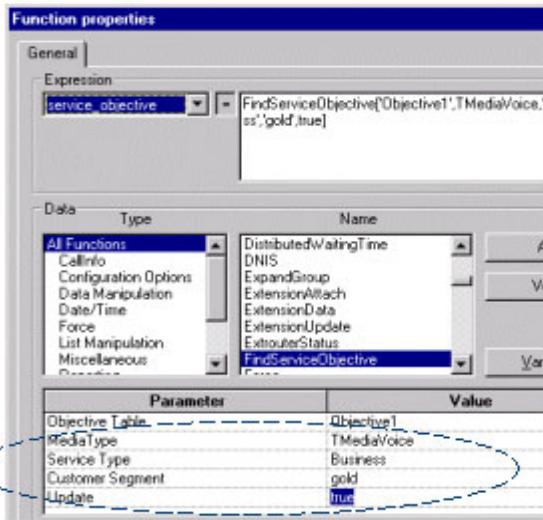
In the case of an unsuccessful search, this function does not use the default triplet for a specified Media Type, but instead returns `0` and generates error code `0018 Unknown object`.

Attaching Business Attributes to Interactions

Use this function to attach `Business Attributes` to interactions as follows:

1. As the strategy developer, associate each interaction with predefined `Business Attributes` in Configuration Layer, such as `Customer Segment`, `Service Type`, `Media Type`, `Service Objective`.
2. Set the `Update` parameter in `FindServiceObjective` to `true` so that URS automatically attaches the `Business Attributes` to each interaction. Or use the `MultiAttach` (see [page 140](#)) object for this purpose.
URS attaches default data to interactions not associated with any `Business Attributes`. Default data can be predefined per `Media Type` in Configuration Layer.
3. Set the option `report_targets` (see [page 660](#)) to `true` to expose the data.

[Figure 219](#) shows an example properties dialog box and attached data for `FindServiceObjective`.



Business Attribute data attached to interaction:

```
AttributeUserData [79] 00 02 00 00..
  'CustomerSegment' 'gold'
  'ServiceType' 'Business'
  'ServiceObjective' '30'
```

Default data attached to voice interactions:

```
AttributeUserData [79] 00 02 00 00..
  'CustomerSegment' 'default'
  'ServiceType' 'default'
  'ServiceObjective' '20'
```

Figure 219: Using FindServiceObjective to Attach Data

ListGetDataCfg

Parameters: List: STRING (List object name, constant or variable)

Item: STRING (constant or variable)

Key: STRING (constant or variable)

Return value type: STRING

An IRD list object (see [page 73](#)) contains strings of any nature (for example, DNIS or ANI strings). May be used to create lists of toll-free numbers. Rather than reference each individual 800 number in a strategy, you can logically group numbers together and name the group. Then, when you need to add/edit numbers, the strategy does not need changing; you just add to/edit the list.

This function looks for an element Item in the list configuration object and return the value of its property Key. The name of the list is provided in the

parameter List. If the member is not found or if it doesn't have the property, the function returns empty string.

Starting with release 8.0, if the ListGetDataCfg function is used to get all the properties of the List element in the format: key1:value1|key2:value2| and so on, the key value '*' must be used. The previous method (available in 7.x) to get the list of all properties by using an empty key is deprecated.

If the URS option function_compatibility is set to a value other than 8.0, the old method (empty parameter) works as it did prior to the 8.0 release.

ListLookup

Parameters: List: STRING, Table Access, or variable representing a string for the list

Value: STRING or variable (representing a string for the value)

Return value type: INTEGER

ListLookup offers quick limited access to stored data without the delay of an actual database lookup. This function checks whether the string value appears in the result of a database inquiry specified in a Table Access object in the Configuration Layer. The name of the Table Access object is provided in the parameter List. The returned value is 1 (true), if Value is a comma-delimited substring of the inquiry result; otherwise, the returned value is 0 (false.)

When selecting a List, the value can be specified in two ways:

- Enter the string manually.
- From the dialog box, select All Functions (and then the function name), or select Interaction Data or Variables, and then select or enter a value. The values available are for the configured Table Access. The values available for the Variable type are the variables included in the Variable list.

Note: See “Specifying Variables in Dialog Boxes” on [page 430](#) for information on variable usage for function parameters.

The following error messages are possible:

- 0013 Remote error—error/unknown message from DB Server
- 0018 Unknown object—unable to find specified Table Access object

User-Defined Table Access Objects

Starting with Universal Routing 7.x, the ListLookup function works with Table Access objects of the user defined type. To see how this works:

1. Create a Table Access of type ANI. This Table Access must reference a Format with one column Field of type ANI.

Note: You can create a Table Access of any type, but it must reference a Format with a Field defined with a matching type.

2. Give the field a char data type and specify a length that matches the length specified for the column.
3. Define the Table Access as either cacheable or non-cacheable. If cacheable, you can also use the Updateable property to specify at what point the cached data should be refreshed. If there is plenty of memory on the URS machine and the cached data is not too large, cache it. If not, then when the ListLookup function executes, there will be a delay in strategy execution in order to send the request to the DB Server and database and get the result.
4. Define the second parameter of the ListLookup function, which is a value (you can use a variable). The function will use the Table Access to retrieve a comma-separated list of values (all values in the one column Field specified in the Format.) Once retrieved, you have the option of caching the values. From this list, it will look for a value that matches whatever you specified as the second parameter. It returns 1 (true) if it finds it, 0 (false) if it doesn't.

ListLookupCfg

Parameters: List: STRING (List object name, constant or variable)
 Value: STRING or variable (representing a string for the value)

Return value type: STRING (constant or variable)

Note: See “List Objects” on [page 73](#).

This function checks whether the element Member is member of a List configuration object. The name of the List is provided in the parameter list. The returned value is 1 (true) if Value is member of List. Otherwise, the returned value is 0 (false.)

This function behaves exactly like the ListLookup function except that what is scanned is not some table from the customer database, but a List object from the Configuration Database.

Date and Time Functions

The IRD Function Properties dialog box with Date/Time selected is shown in Figure 220.

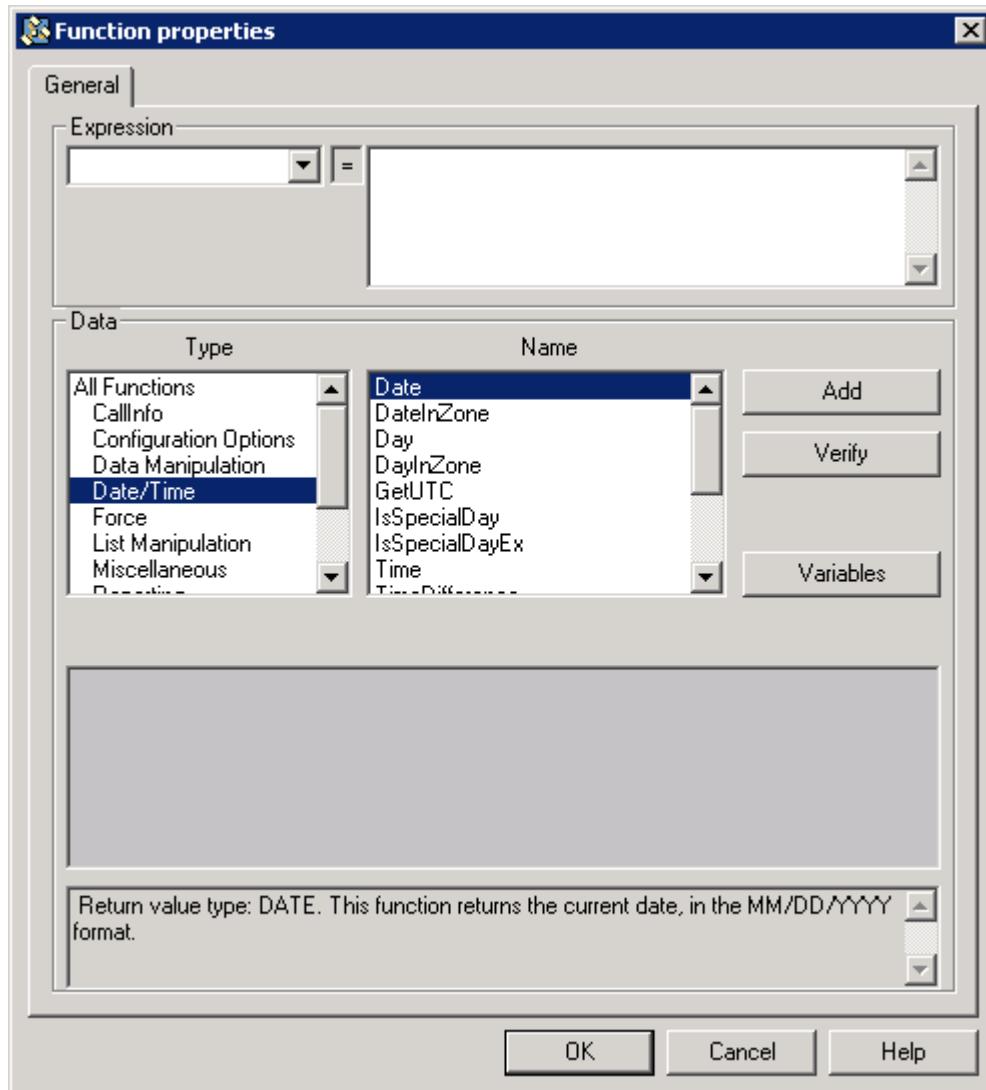


Figure 220: Function Properties Dialog Box, Date/Time Functions

This section presents functions in the Date and Time category (see Table 130 on [page 430](#)).

Date

Parameters: none

Return value type: DATE

This function returns the current local date (date in the time zone where URS is running); that is, the number of days since January 1st 1970. See DATE value type description in Table 128 on [page 425](#). The result of this function can be compared to the dates in the format MM/DD/YYYY or MM/DD/ (the current year is implied). Example: Date []<01/01/2001.

DateInZone

Parameter: TimeZone: STRING or variable (representing a string for the time zone)

Return value type: DATE

Returns the current date in the specified time zone; that is, the number of days since 1 January 1970 (January, 1970, is day 1). The result of this function can be compared to the dates in the format MM/DD/YYYY or MM/DD/ (the current year is implied). Example: DateInZone [PST]<01/01/2001.

The TimeZone parameter is the name of a time zone configured in Configuration Layer.

When selecting a time zone, the value of the TimeZone parameter can be specified in two ways:

- Enter the time zone manually, as configured in the Configuration Layer.
- From the dialog box, select All Functions (and then the function name), or click under Value and select a Time Zone, or click Variables and select or enter a value. The values available are for the configured time zones. The values available for the Variable type are the variables included in the Variable list.

The following error message is possible:

- 0018 Unknown object—unable to find the TimeZone in the Configuration Database

Day

Parameters: none

Return value type: DAY

This function returns the current local day of the week (day in time zone where URS is running); that is, a number from the range of 0-6. See DAY value type description in Table 128 on [page 425](#). The result of this function can be compared to the named constants for days of the week (Sunday=0, Monday=1, Tuesday=2, Wednesday=3, Thursday=4, Friday=5, or Saturday=6). Example: Day []<Friday.

DayInZone

Parameters: TimeZone: STRING or variable (representing a string for the time zone)

Return value type: DAY

This function returns the current day of the week in the specified time zone. The result of this function can be compared to the named constants for days of the week. Example: DayInZone[PST]<Friday.

The TimeZone parameter is the name of a time zone configured in Configuration Layer.

When selecting a time zone, the value of the TimeZone parameter can be specified in two ways:

- Enter the time zone manually, as configured in the Configuration Layer.
- From the dialog box, select All Functions (and then the function name), or click under Value and select a Time Zone, or click Variables and select or enter a value. The values available for are the configured time zones. The values available for the Variable type are the variables included in the Variable list.

The following error message is possible:

- 0018 Unknown object—unable to find the TimeZone in the Configuration Database

GetUTC

Parameters: none

Return value type: INTEGER

Note: Since radio signals can cross multiple time zones and the international date line, some worldwide standard for time and date is needed. This standard is coordinated universal time, abbreviated UTC. This was formerly known as Greenwich mean time (GMT).

This function returns UTC in seconds, which is the number of seconds elapsed since midnight (00:00:00), January 1, 1970, according to the system clock.

Use this function when setting the scheduled time for submitting interactions in queues to routing strategies in IRD's Interaction Design window as described in the *Universal Routing 7.6 Business Process User's Guide*.

Specifically, see the section on View object configuration, Scheduling tab, in the chapter on creating business process objects.

Use this function with the Assign object (see [page 120](#)) or directly as a value of some other function parameter. Below is an example to get a time string for scheduling some action in exactly 24 hours where t is the name of a variable.
Assign[t, GetUTC[]]

IsSpecialDay

Note: IsSpecialDayEx on page 488 provides the same functionality as IsSpecialDay, but includes additional parameters.

Parameters: Stat Table: STRING or variable (representing a string for the Stat Table)

Stat Day: STRING or variable (representing a string for the Stat Day)

Return value type: INTEGER

Use to base strategy actions on the existence of holidays, special closing times, or other types of special days. The returned result is interpreted as a Boolean value, where value 0 indicates false and value 1 indicates true.

The Stat Day parameter of this function is optional. If it is specified, URS inquires from Configuration Server whether the specified Statistical Day is configured for the specified Statistical Table and whether the current date meets the definition of the Statistical Day.

If the Stat Day parameter is set to the empty string (two single quotation marks with no space between them), URS inquires from Configuration Server whether the current date meets the definition of any of the Statistical Days configured for the specified Statistical Table.

When selecting a Stat Table or Stat Day, the value can be specified in two ways:

- Enter the string manually.
- From the dialog box, click under Value. In the resulting Key dialog box, select a Type (Stat Table or Stat Day), or Variable, and then select or enter a value. The values available are for configured Stat Tables or Stat Days. The values available for the Variable type are the variables included in the Variable list.

There is one possible error message:

- 0018 Unknown object—unable to find specified Stat Day or Stat Table in the Configuration Database

IsSpecialDayEx

Parameters: Stat Table: STRING or variable (representing a string for the Stat Table)

Stat Day: STRING or variable (representing a string for the Stat Day)

Time Zone: Select a time zone or use a variable (representing a string for the time zone)

UseTime: Select a value of true or false (default).

Return value type: INTEGER

The returned result is interpreted as a Boolean value, where 0 indicates false and 1 indicates true.

This function works the same as IsSpecialDay, but offers two additional parameters: Time Zone and UseTime. UseTime specifies whether or not to use the time limits within a Stat Day.

A non-empty value for the Stat Day parameter is optional. If Stat Day is specified, URS inquires from Configuration Server:

Whether the specified Stat Day is configured for the specified Stat Table and

whether the current date and current time (if UseTime is true) meet the definition of the Statistical Day.

If Time Zone is not empty, then current date and time are taken in this TimeZone.

Time

Parameters: none**Return value type:** TIME

This function returns the current local time (time in the time zone where URS is running); that is, the number of minutes from midnight (00:00 AM). See TIME value type description in Table 128 on [page 425](#). The result of this function can be compared to time values in the format hh:mm [AM/PM].

Example: `Time[]<11:00 AM`

TimeDifference

Parameters: First Time: TimeStamp

Second Time: TimeStamp

Return value type: INTEGER

The returned value is the result of subtracting the First Time parameter from the Second Time parameter.

First Time and Second Time are thought of as moments specified in milliseconds. If First Time is greater than Second Time, it is presumed that Second Time occurs on the day following First Time. Therefore, 24 hours (86,400,000 milliseconds) are added to Second Time before the subtraction.

This TimeStamp value refers to the TimeStamp function ([page 490](#)) described below. It is recommended that the TimeDifference function not be used for different days.

TimeInZone

Parameters: `TimeZone`: STRING or variable (representing a string for the time zone)

Return value type: TIME

This function returns the number of minutes elapsed since the last midnight (00:00 AM) in the specified time zone. The result of this function can be compared to time values in the format hh:mm. The `TimeZone` parameter is the name of a time zone configured in Configuration Layer. For example:

`TimeInZone[PST]<11:00.`

When selecting a time zone, the value can be specified in two ways:

- Enter the time zone manually.
- From the dialog box, select All Functions (and then the function name), or click under Value and select a Time Zone, or click Variables and select or enter a value. The values available are for the configured Time Zone. The values available for the Variable type are the variables included in the Variable list.

The following error message is possible:

- `0018 Unknown object`—unable to find the `TimeZone` in the Configuration Database

TimeStamp

Parameters: none

Return value type: INTEGER

The result of this function is the current time in milliseconds, measured from midnight the same day.

UTCAdd

Parameters: `UTC` representing coordinated universal time: STRING. For information on UTC, see the note on [page 487](#).

`Years` representing the number of years to increment: INTEGER

`Months` representing number of months to increment: INTEGER

`Days` representing number of days to increment: INTEGER

`Hours` representing number of hours to increment: INTEGER

`Minutes` representing number of minutes to increment: INTEGER

`Seconds` representing number of seconds to increment: INTEGER

Return value type: INTEGER

This function increments the provided universal coordinated time (UTC) with the specified number of years, months, and so on (the numbers can be negative).

Use this function when setting the scheduled time for submitting interactions in queues to routing strategies in IRD's *Interaction Design* window as described in the *Universal Routing 7.6 Business Process User's Guide*.

Specifically, see the section on View object configuration, *Scheduling* tab, in the chapter on creating business process objects.

Use this function with the Assign object (see [page 120](#)) or directly as a value of some other function parameter. Sample:

```
Assign[t, UTCAdd[t, 0, 0, 1, 0, 0, 0]] or Assign[t, UTCAdd[t, 0, 0, 0, 24, 0, 0]]
```

The View object *Scheduling* tab description in the *Universal Routing 7.6 Business Process User's Guide* contains a sample strategy that uses this function.

Note: If the UTC time input is less than zero, in this case, 0 (zero) will be used. Addition/subtraction is performed in the following order: years, months, days, hours, minutes, seconds.

UTCFromString

Parameters: `UTC String`: STRING or variable (representing a string for the time zone in coordinated universal time format). For information on UTC, see the note on [page 487](#).

Return value type: INTEGER

This function takes a UTC string in the format `YYYY-MM-DDTHH:MM:SSZ` and returns it as the number of seconds.

Use this function when setting the scheduled time for submitting interactions in queues to routing strategies in IRD's *Interaction Design* window as described in the *Universal Routing 7.6 Business Process User's Guide*.

Specifically, see the section on View object configuration, *Scheduling* tab, in the chapter on creating business process objects.

Use this function with the Assign object (see [page 120](#)) or directly as a value of some other function parameter. Example (where `s` is the name of a variable).

```
Assign[s, UTCFromString[t]]
```

The View object *Scheduling* tab description in the *Universal Routing 7.6 Business Process User's Guide* contains a sample strategy that uses this function.

Warning! If you provide the string in the wrong format, the returned value will be unpredictable.

UTCToString

Parameters: UTC representing coordinated universal time: INTEGER. For information on UTC, see the note on [page 487](#).

Return value type: STRING

This function takes coordinated universal time (UTC) in seconds and converts it into a string with the format YYYY-MM-DDTHH:MM:SSZ.

Use this function when setting the scheduled time for submitting interactions in queues to routing strategies in IRD's Interaction Design window as described in the *Universal Routing 7.6 Business Process User's Guide*.

Specifically, see the section on View object configuration, Scheduling tab, in the chapter on creating business process objects.

Use this function with the Assign object (see [page 120](#)) or directly as a value of some other function parameter. Example:

```
Attach['ScheduledAt', UTCToString[·]]
```

The View object Scheduling tab description in the *Universal Routing 7.6 Business Process User's Guide* contains a sample strategy that uses this function.

Note: If UTC input is less than zero, then 0 (zero) will be used.

Force Functions

[Figure 221](#) shows the Function Properties dialog box with Force selected.

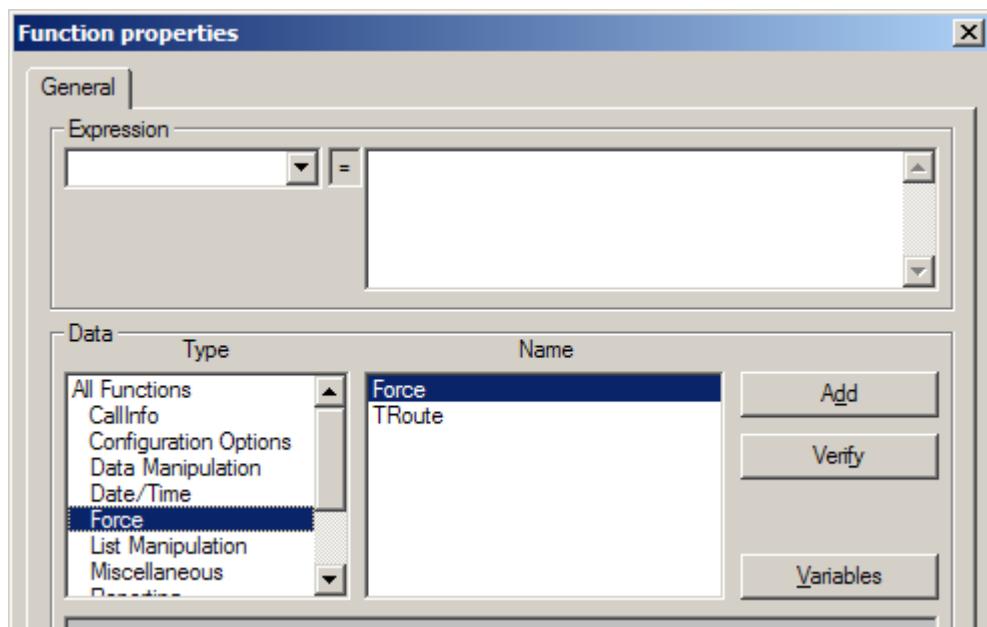


Figure 221: Function Properties Dialog Box, Force Functions

This section presents functions in the Force category (see Table 130 on [page 430](#)).

Force

Parameters: none

Return value type: VOID

This function instructs URS to perform forced interaction routing, under which calls or multimedia interactions are routed regardless of the target's current state. Targets for forced routing must have a format of `DN@Switchname` or merely `DN`. (The latter format will route the interaction to the corresponding DN on the current switch.) The target is ignored if assigned a type. For multimedia interactions, the target Agent or Place must be the only target for the interaction.

The target to which the interaction will be forcibly routed is one of the targets listed in the properties of the first Routing object following the Force object. After the Force object, it is advisable to place only a single Routing object.

The effect of the Force object cannot be annulled after the function is called. If, under some conditions, ordinary routing still needs to be performed after Force is invoked, use a combination of the functions SelectDN, SuspendForDN, and RouteCall, described on pages [608](#), [614](#), and [607](#), respectively.

Note: Force was retained in URS for the sake of compatibility with earlier versions. When building new strategies, use the forced-routing function TRoute, described on [page 493](#), since the explicit specification of the parameters of TRoute makes its operation more transparent and decreases the risk of errors.

TRoute

Parameters:

Destination: STRING or variable (representing a string for the destination)

Location: STRING or variable (representing a string for the destination)

Route Type: `RouteTypeUnknown`, `RouteTypeDefault`,
`RouteTypeLabel`, `RouteTypeOverwrittenDNIS`, `RouteTypeDDD`,
`RouteTypeIDDD`, `RouteTypeDirect`, `RouteTypeReject`,
`RouteTypeAnnouncement`, `RouteTypePostFeature`,
`RouteTypeDirectAgent`, `RouteTypePriority`,
`RouteTypeDirectPriority`, or variable

DNIS: STRING or variable (representing a string for the destination)

Return value type: VOID

This function instructs URS to send a routing instruction to T-Server with the specified parameters. This function is recommended as a better alternative to Force.

The following errors are possible:

- 0001 Unknown error
- 0008 Routing done
- 0013 Remote error

Note: When executing a strategy where force (TRoute function) or default routing is applied to an interaction, URS removes the call from all virtual queues and only then executes the force or default routing.

List Manipulation Functions

The name of this category of functions comes from the fact that all the function in it work with strings in list format.

The IRD Function Properties dialog box with List Manipulation selected is shown in [Figure 222](#).

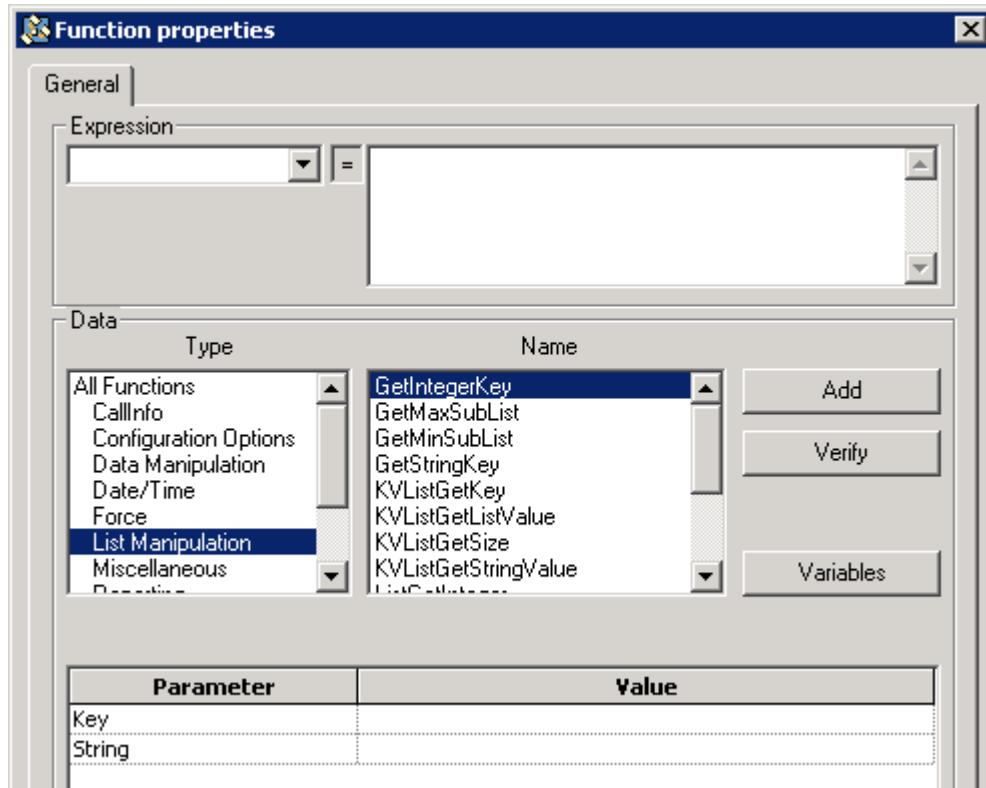


Figure 222: Function Properties Dialog Box, List Manipulation Functions

This section presents functions in the List Manipulation category (see Table 130 on [page 430](#)).

GetIntegerKey

Parameters: Key: STRING or variable (representing a string for the key)
String: STRING (list) or variable (representing a string)

Return value type: INTEGER

This function parses out an integer value for a specified key from a key-value list. This value is obtained from the leading sequence of digits before the first nondigit character in the value. If the value starts with a character other than a digit, or if the searched-for key does not appear in the list, the function returns value 0. If the key is repeated in a list, its first occurrence is retrieved.

GetMaxSubList

Parameters: List: STRING (list) or variable (representing a string for the list)

Return value type: STRING (list)

Parses out key-value pair(s) for pairs with the maximum value. All values are interpreted as integers.

For example, if str is key1:5|key2:4|key3:5, then GetMaxSubList(str) returns key1:5|key3:5.

GetMinSubList

Parameters: List: STRING (list) or variable (representing a string for the list)

Return value type: STRING (list)

Parses out key-value pair(s) for pairs with the minimum value. All values are interpreted as integers. For example, if str is key1:5|key2:4|key3:5, then GetMinSubList(str) returns key2:4.

GetStringKey

Parameters: Key: STRING or variable (representing a string for the key)
String: STRING (list) or variable (representing a string)

Return value type: STRING

Parses out the value for a specified key from a key-value list. If the searched-for key does not appear in the list, the function returns the empty string. If the key is repeated in the list, its first occurrence is retrieved.

GetRawAttribute

Parameters: Key: STRING or variable (representing a string for the key)
 String: STRING (list) or variable (representing a string)

Return value type: STRING

This function returns the value of a specific attribute (or empty string if there is no such attribute). You must obtain the numeric values of attributes for which you are using this function (see the sample below). Events, such as `EventRouteRequest` have a small subset only of these attributes, and some never appear as attributes. For example, the attributes for `EventRouteCall`. However, URS expects that the snapshot of an event can contain any set of attributes.

Limitations

The following limitations exist when this function is used:

- URS is enabled with the `tattributes` option (in `default` section) with a value that contains a comma-separated list of attribute numbers. For example, `tattributes="10, 24, 47"`. The option is not dynamically updated, but is static. A snapshot is taken of those attributes only that are listed as values of this option and accessed by the `GetRawAttribute` function. By default, the value of this option is empty and no snapshot is taken of any attribute.
- URS does not snapshot attributes whose values are binary or KVLists. (It snapshots strings, integers, or ConnectionIDs only.)

Sample

This sample shows a partial list of attributes only.

To access `AttributeNetworkNodeID` (numeric value 6, see below) from the strategy, URS must have option `tattributes="6"` then, `GetRawAttribute[6]` returns the required value.

```
enum TAttribute {
    AttributeProtocolVersion,      /* 0 */
    AttributeErrorMessage,
    AttributeReferenceID,
    AttributeCallID,
    AttributeNodeID,
    AttributeNetworkCallID,
    AttributeNetworkNodeID,
    AttributeTransferredNetworkCallID,
    AttributeTransferredNetworkNodeID,
    AttributeConnID,
    AttributePreviousConnID,      /* 10 */
```

```

AttributeTransferConnID,
AttributeConferenceConnID,
AttributeCallState,
AttributeAgentID,
AttributeAgentStateReasonUnused,
AttributeAgentWorkMode,
AttributeReason,
AttributeDNIS,
AttributeANI,
AttributeThisDN,           /* 20 */
AttributeThisQueue,
AttributeThisTrunk,

```

KVList Functions

These functions work with the LIST data type, providing read access to LIST properties used to extract internal data within an obtained LIST object. LIST variables can be used to accept output of the Find Interactions object. Every LIST object represents and is internally implemented as a regular KVList. Data in the format of a KVList are returned by such strategy objects as Web Service and External Services.

KVListFindSubList

Parameters: List: LIST or variable presenting a collection of smaller sublists.
Properties: LIST or variable (defining the search criteria for sublists).

Return value type: LIST

This function presents an extension of an existing set of KVList functions. It finds and returns the first sublist inside the input list that satisfies the search criteria. The search criteria is a subset of the properties that the sublists could have.

KVListGetKey

Parameters: Index: INTEGER or variable
List: LIST or variable (having LIST object as value)

Return value type: STRING

Returns the name of the key in the key-value pair, with sequential number Index starting from 1. If the key-value pair with that name does not exist, an empty string is returned.

KVListGetValue

Parameters: List: LIST
Key: STRING (key)

Return value type: LIST

It is assumed that the provided List has a key-value pair with name Key and a nested sublist as a value. This function returns this sublist. If no such pair exists, an empty LIST object will be returned.

For example, if the Find Interactions object stores the result of a variable Result and goes through all found interactions, this logic can be used:

```
Interactions= KVListGetValue(Result, 'Interactions');
N= KVListGetSize(Interactions);
for n from 1 to N {
    InteractionID= KVListGetKey(Interactions, n);
    InteractionProperties= KVListGetValue(Interactions,
InteractionID);
    Property1= KVListGetStringValue(InteractionProperties,
'SomePropertyName');
    .
    <do whatever you like with this interaction's properties>
    .
}
```

KVListGetSize

Parameters: List: LIST

Return value type: INTEGER

Returns the number of top-level key-value pairs in the List.

KVListGetStringValue

Parameters: List: LIST
Key: STRING (key)

Return value type: STRING

It is assumed that the provided List has a key-value pair with name Key. This function returns the value of this pair in string format. If no such pair exists, an empty string will be returned.

ListGetInteger

Parameters: Index: INTEGER or variable
List: STRING (list) or variable (representing a string for a list)

Return value type: INTEGER

Parses out the integer value for the specified key from a key-value list based on the position, starting from 1, rather than the key.

For example, if `str` is `key1:3|key2:value2|key3:value3`, then `ListGetInteger(1, str)` returns 3 for key1. `ListGetInteger(3, str)` in comparison, returns 0 because `value3` is not converted into an integer. Any index greater than the number of key-value pairs returns 0, which means the Index is out of bounds.

ListGetKey

Parameters: Index: INTEGER or variable

List: STRING (list) or variable (representing a string for a list)

Return value type: STRING

Parses out the key from a key-value list based on the position (int index - integer index), starting from 1, rather than the key.

For example, if `str` is `key1:value1|key2:value2|key3:value3`, then `ListGetKey(3, str)` returns key3. Any index greater than the number of key-value pairs returns “” (an empty string), which means the Index is out of bounds.

ListGetSize

Parameters: List: STRING (list) or variable (representing a string for the list)
IgnoreEmptyItems (true or false)**Return value type:** INTEGER

Returns the number of key-value pairs.

For example, if `str` is `key1:value1|key2:value2|key3:value3`, then `ListGetSize(str)` returns 3.

URS supports empty key-value pairs in List object key-value strings. As a result, URS does not distinguish an empty List object (one having no key-value pairs) from a List object that has one empty key-value pair. Since both are presented as empty strings, URS considers them both as valid List objects with one empty key-value pair. Accordingly, the `ListGetSize` function returns 1 for such a List object. In cases where this is not desirable, the `ListGetSize` function has a second parameter that specifies whether URS should take empty key-value pairs into account.

ListGetString

Parameters: Index: INTEGER or variable

List: STRING (list) or variable (representing a string for a list)

Return value type: STRING

Parses out the string value for the specified key from a key-value list based on the position, starting from 1, rather than the key.

For example, if the list is `key1:value1|key2:value2|key3:value3`, then `ListGetString(3, list)` returns the `value3` for `key3`. Any Index greater than the number of key-value pairs returns “” (an empty string), which means the Index is out of bounds.

SetIntegerKey and SetStringKey

These functions allow you to easily create and modify key-value lists without keeping track of separators and spaces. They also ensure that the resulting key-value lists have no repeated keys.

SetIntegerKey

Parameters: `Key`: STRING or variable (representing a string for the key)
`Value`: INTEGER or variable
`List`: STRING (list) or variable (representing a string for the list)

Return value type: STRING (list)

This function returns a key-value list obtained from List by concatenating to it the key-value pair specified by the first two arguments and erasing any already existing pairs with the same key.

SetStringKey

Parameters: `Key`: STRING or variable (representing a string for the key)
`Value`: STRING or variable (representing a string for the value)
`List`: STRING (list) or variable (representing a string for the list)

Return value type: STRING (list)

This function returns a key-value list obtained from List by concatenating to it the key-value pair specified by the first two arguments and erasing any already existing pairs with the same key.

TargetState

Parameters: `Target`: STRING (statistical object) or variable (representing a string for the target Agent or Place).

Return value type: LIST

This function accepts standard target specifications, however, the target type must be either agent or place. No other types of target are supported. URS returns all known information about this target, such as employee ID, place name, state of target, number of available DNs, and total number of DNs

owned by the target. It also returns the following additional information for every DN: number, switch name, status, type, readiness, and other data.

Note: In multimedia-enabled cases (when agent capacity is configured) the media channels, such as email and chat are also considered to be DNs.

Sample TargetState Function	UN_105_vit_sw2@vit-statserver.A
Result	"status:4 agent:UN_105_vit_sw2 place:Place_105_vit_sw2 login:105 loadin g:0 ready:1 voice:1 dnsrdy:2 dnsall:2 dn.#1.type:2 dn.#1.number:105 dn. #1.switch:vit_sw2 dn.#1.status:4 dn.#1.loading:0 dn.#1.ready:1 dn.#1.me dia:0 dn.#2.type:1 dn.#2.number:115 dn.#2.switch:vit_sw2 dn.#2.status:4 dn.#2.loading:0 dn.#2.ready:1 dn.#2.media:0"

All of the data for agent or place can be extracted and analyzed by using the List Manipulation function. [Table 133](#) contains a description of the fields in the returned LIST.

Table 133: TargetState Function results—LIST Fields

Field	Description
status:4	The agent voice status exactly as reported by statserver.
agent:UN_105_vit_sw2	The employee ID of the agent.
place:Place_105_vit_sw2	The place where the agent is logged in.
login:105	The login that is used by the agent to log in to this place.
loading:0	The current agent loading. (This value is returned by the StatAgentLoading.
dn.#1.type:2	The first DNs type (same type as the TargetState function).
ready:1	The agent is considered (by URS) to be ready.
voice:1	The agent is considered (by URS) to be able to accept voice calls. (The agent has at least one ready voice DN or its voice capacity is positive.)
dnsrdy:2	The number of ready DNs.
dnsall:2	The total number of DNs.
dn.#1.type:2	The first DN with the type as reported by Stat Server.

Table 133: TargetState Function results—LIST Fields (Continued)

Field	Description
dn.#1.number:105	The first DN and its number.
dn.#1.switch:vit_sw2	The first DN and its switch.
dn.#1.status:4	The first DN and its status as reported by stat Server.
dn#1.loading:0	The first DN and its loading state.
dn.#1.ready:1	The first DN and its ready state.
dn.#1.media:0	The first DN and whether or not it has media.
dn.#2.type:1	The second DN with the type as reported by Stat Server.
dn.#2.number:115	The second DN and its number.
dn.#2.switch:vit_sw2	The second DN and its switch.
dn.#2.status:4	The second DN and its status as reported by stat Server.
dn.#2.loading:0	The second DN and its loading state.
dn.#2.ready:1	The second DN and its ready state.
dn.#2.media:0	The second DN and whether or not it has media.
Note: The values in the ready, voice, and media fields are Boolean values, where 1 or 0 means true or false, respectively.	

The numeric Agent and DN status values can be interpreted, based on the Statistics definition dialog that is available in IRD. The list of statuses (in the Mask list control) for Agent and Group statistics can be Agent and DN statuses. For example, starting from 0, NotMonitored=0, Monitored=1, and so on.

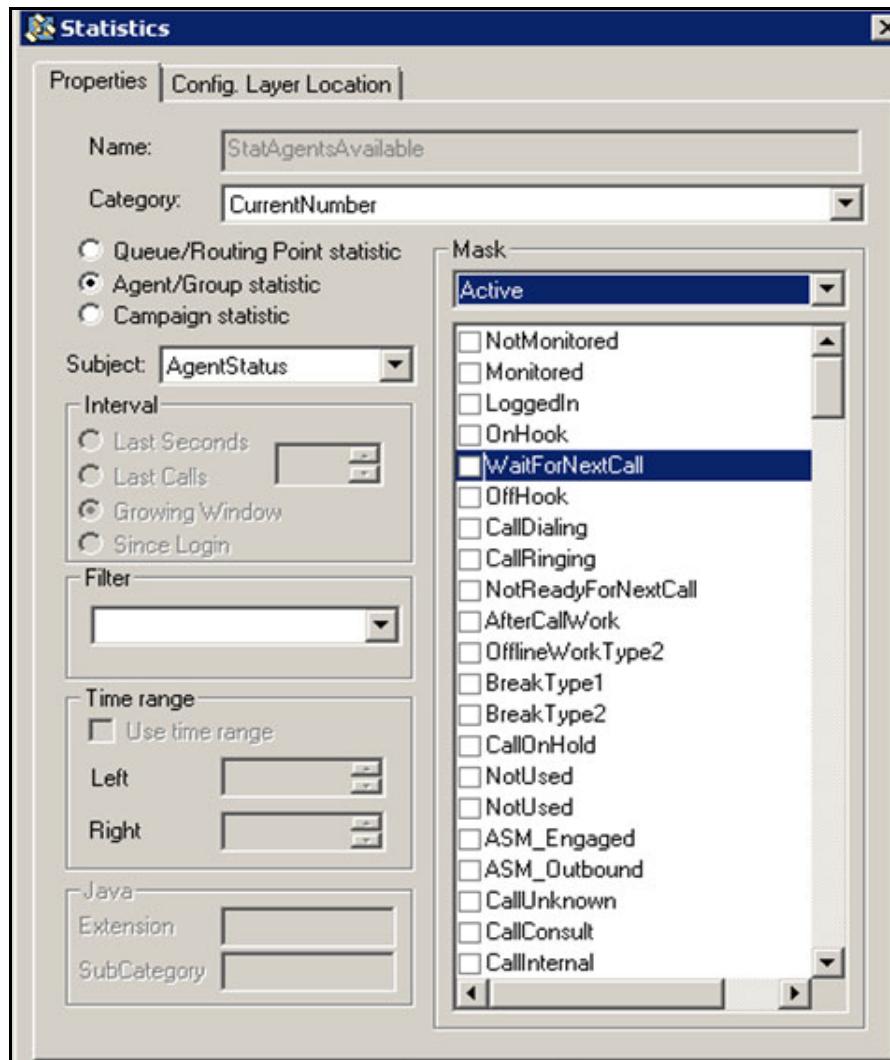


Figure 223: IRD Statistics, Agent Status

The values in the type field can be found in the *Configuration Layer Release 8.0.0 API Reference*. See enumerator CfgDNTYPE. They can also be found in the IRD. See the CME settings folder in the Routing Design Options dialog for the values of CFGDN node tree. For example, values started from 1, Extension=1, ACDPosition=2, and so on. See [Figure 224](#).

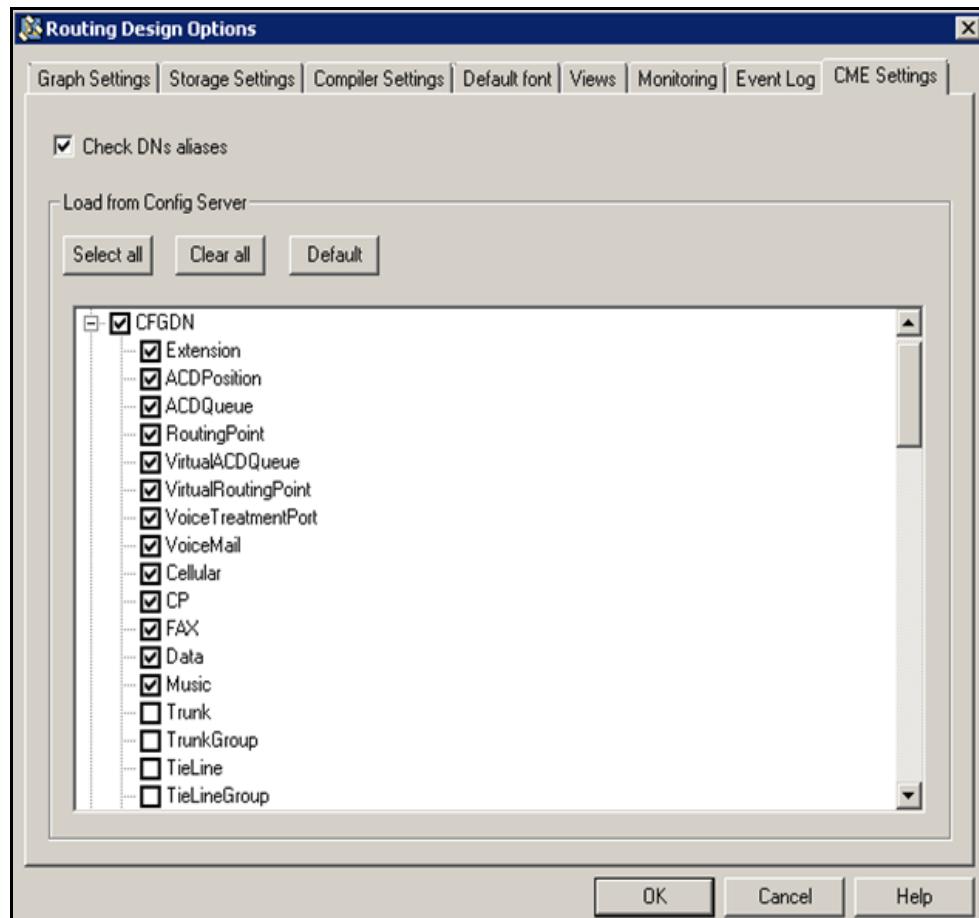


Figure 224: Routing Design Options, DN Aliases

Miscellaneous Functions

Figure 225 on [page 505](#) shows the Function Properties dialog box with Miscellaneous selected.

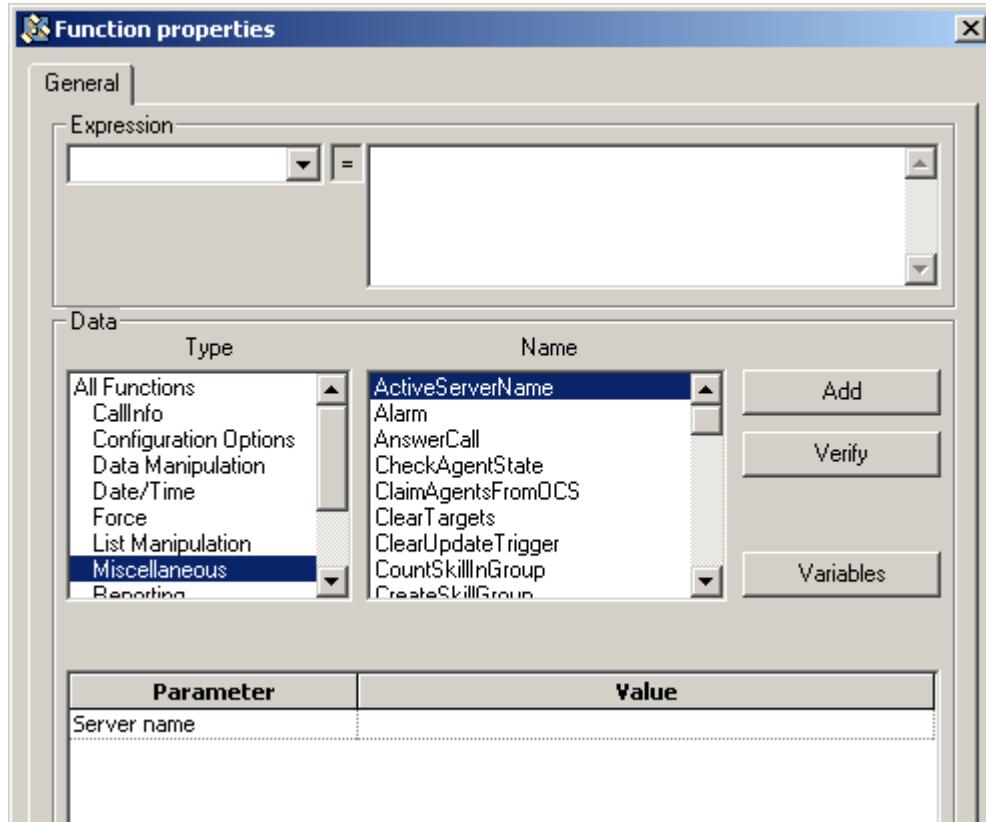


Figure 225: Function Properties Dialog Box, Miscellaneous Functions

This section presents functions in the Miscellaneous category (see Table 130 on [page 430](#)).

ActiveServerName

Parameters: Server Name: STRING, constant or variable (name of primary server)

Return value type: STRING

This function detects and returns the name of the server from the primary/backup pair of servers that URS is actually working with (active) at the moment when function is called.

Use Case

Assume two T-Servers are working in Hot Standby mode with two independent links to different URSSs. Due to routing strategy specifics (play announcement first, then route) different links must use different DN numbers when routing calls.

- In Warm Standby mode, it is possible to find out which link (URS) and which T-Server is primary at the moment if you use the GetCurrentTserver function (see [page 458](#)) in the strategy.
- In Hot Standby mode, this function always returns the name of T-Server that was connected first. This makes it impossible to make a routing decision based on which T-Server is Primary (in other words, which link is going to receive the route request).

Use the ActiveServerName function to return the name of the active T-Server from the pair working in Hot Standby or Warm Standby mode.

Alarm

Parameters: Alarm Number: INTEGER or variable

Alarm Message: STRING or variable (representing a string)

Return value type: VOID

This function sends an alarm message through the URS management API. This message can be used, for example, by the Genesys SNMP Master Agent as a client of the URS management API to generate a polling trap. For more information on setting alarms, see the *Framework 8.1 Deployment Guide*, Management Layer Functionality chapter. Also see the *Framework 8.1 Management Layer User's Guide*, SNMP Interface chapter.

The alarm is sent along with the following information: the supplied Alarm Number, the Alarm Message, and the current interaction's ANI, CED, DN, and Tenant.

AnswerCall

Parameters: none

Return value type: STRING (target)

This function attempts to answer the current interaction.

The returned value can be either `return:ok` or `return:error`.

The function first checks whether the operation is possible. This verification can result in the following errors that can be retrieved by using the Error object immediately after the AnswerCall function:

- `0001 Unknown error.`

- **0002 Transfer in progress**—another transfer is in progress; this error message will be returned if URS has sent an instruction to T-Server to transfer the interaction between two DNs and if, at the time AnswerCall is invoked, no notification is received by URS about the interaction reaching a DN after the transfer. This can only be possible because treatments are running while the strategy executes the function.
- **0003 Treatment in progress**—another treatment is in progress; this error will occur if AnswerCall is invoked after URS has issued an instruction for T-Server to start a treatment but before T-Server responds with EventTreatmentApplied, EventTreatmentNotApplied, or an error.
- **0004 Call is on hold**—the agent puts the interaction on hold; this error message will be returned either if the interaction was put on hold at a DN for which URS is registered or if the interaction cannot be found on any of the DNs for which URS is registered.
- **0005 Call gone**—URS has not registered a DN belonging to the interaction.
- **0008 Routing done**—not permitted in postrouting; this message will be returned if the interaction has already been routed successfully by the function RouteCall or by a Routing object, or if the interaction has been force-routed either by the function TRoute, or by combining the Force object with a Routing object.
- **0013 Remote error**—T-Server reports an error.

If none of the above errors are found, the function proceeds as follows:

1. If the interaction is at a routing point, a virtual routing point, or at a Service Number, the function returns return:ok .
2. If the interaction is at a DN for which EventEstablished has already been received, the function returns return:ok .
3. If neither of the above holds, the function instructs T-Server to answer the interaction. If T-Server returns an error because of T-Server or switch-specific restrictions, the function returns the error message [xxxx] TServer error message, where xxxx is the T-Server error code. If T-Server returns EventEstablished, the function returns return:ok .

CheckAgentState

Parameter: CheckFlag: INTEGER (constant or variable)

Return value type: VOID

This function instructs URS whether to take into account the state of an Agent, Place, Agent Group, or Place Group as reported by Stat Server or to look only for free DNs belonging to the Agent, Place, or Agent Group.

- For the CheckFlag parameter in IRD, choose either Constant type or Variable type.
- For the Constant type, select a value of true or false.

- For the Variable type, select a variable (previously declared in the `Variable list` dialog box).

Important Information

- `CheckAgentState[false]` makes it possible to route a voice interaction to an agent that Stat Server reports as not ready.
- If agent capacity rules are set, the `CheckAgentState` function has no effect (the Genesys Agent Capacity model does not use agent state).
- To allow agent to receive multiple voice interactions, `CheckAgentState` must be false. However, the side effect is that URS distributes interactions so as to occupy all DNs of Agent A before considering Agent B.
- When `CheckAgentState` is set to `false` or the function `UseAgentState` (see [page 554](#)) is used, URS does not apply the `verification_time` option (see [page 686](#)) to agents, but still applies it to agent DNs.
- Also see the option “`use_agent_capacity`” on [page 674](#).

ClaimAgentsFromOCS

Parameters: `OCS`: STRING or variable (representing a string for an Outbound Contact Server)

`Virtual Queue`: STRING or variable (representing a string for the Virtual Queue)

`Explicit Targets`: STRING or variable (representing a comma-separated explicit target list)

`Priority`: INTEGER or variable (priority of claiming request, for details see *Outbound Contact 8.1* documentation)

`Interval`: INTEGER (length of time (in seconds) the agent is claimed)

Return value type: INTEGER

This function is used to request that an Outbound Contact Server (OCS) free some agents who currently are busy with outbound activity so they can be available for processing *inbound* calls. While URS is able to override the current agent assignment through the `UseActivityType` function and use any agent for an inbound call, this practice is not recommended. Instead, use the more graceful negotiation with OCS for agent allocation between inbound and outbound that this `ClaimAgentsFromOCS` function provides.

This function is used when a strategy that is processing inbound calls can not process a call within the service objective, *and* the call is important enough to require the use of agents currently busy with outbound activity. In the simplest case, the outbound agents can be claimed if the call is waiting to be processed longer than some specified value.

The parameters of this function are:

- **OCS.** The name of the Outbound Contact Server to which is sent the request to reallocate the agent(s). This parameter is mandatory and will be used by URS to find the T-Server and communication DN used as the communication channel with the OCS. See the *Outbound Contact Deployment Guide* for information on how to configure Communication DNS.
 - **Virtual Queue.** Provides information about the set of agents to be claimed from OCS. This parameter is used only if the **Explicit Targets** parameter is not provided. **Virtual Queue** narrows the set of agents that URS will consider reassigning to the waiting interaction only to those agents in the virtual queue.
- When using the **Virtual Queue** parameter, the `ClaimAgentsFromOCS` function needs to be called after a call is placed in the waiting queue by some means (for example, after the Target Selection object).
- **Explicit Targets.** A comma-separated list of targets (agents or agent groups) that explicitly specifies the set of agents to consider for claiming. This parameter allows claiming agents for whom a call is not currently waiting. In this case, the `ClaimAgentsFromOCS` function can be used at any point in the strategy.
 - **Priority.** An integer indicating the priority of the claiming request.
 - **Interval.** A value for how long (in seconds) the agent is claimed. If not specified (that is, if set to `0`), then URS will derive the time for how long to claim the agent(s).

URS uses different procedures for creating a list of agents to claim depending on whether the **Explicit Targets** or **Virtual Queue** parameter is used.

URS gets the provided list of targets (either through **Virtual Queue** or through **Explicit Targets**) and removes from it all inappropriate agents. If the refined set of agents contains at least *one* agent, then URS sends a request to OCS; if not, URS does not send a request.

An agent is considered *appropriate* under the following conditions:

- An agent is assigned to some campaign group (agents already working on inbound calls are removed, as there is no use to claim them).
- If **Virtual Queue** is used (no **Explicit Targets**), then the agent also must:
 - be logged in
 - have DN of appropriate type
 - not be excluded (see the `ExcludeAgent` function)
 - satisfy custom states (if any are used) (see the function `Use(CustomAgentPlace/DN)Type`).

The function returns the number of agents contained in the claiming request to OCS. The function returns `0` if the claiming request was not sent (for any reason). In such cases, flow control goes through the red port, and the generated error codes (see the `Error` segmentation object) will be:

`0018 Unknown object – specified OCS was not found`

0014 Unknown server – T-Server or Communication DN was not found for communication with the specified OCS

002 Negative answer – No suitable agents were found to claim from OCS.

Note the following:

- While URS provides a set of agents in the claiming request, the OCS always allocates either none or only one agent per claim. URS just provides the set of agents from which the OCS can select.
- The function does not wait for any confirmation from OCS; it just sends the claiming request and strategy execution is continued.
- Since the function `ClaimAgentsFromOCS` depends on the value of the `CurrentAgentAssignment` statistic (which campaign group the agent is assigned to), this function depends on proper setup of Outbound statistics. This means that the `ClaimAgentsFromOCS` function can fail to send a request to OCS for the very first call (when agent assignment statistics are not yet opened).
- Different calls can issue requests containing the same set of agents. URS does not maintain an association with the claiming call, agents list, or agents reassigned back to inbound activity by OCS. When an agent is assigned to an inbound activity, it is processed by URS without any relation to the call(s) having issued claiming requests. URS will route to such agent the best call it has at the moment (which may or may not be the call that issued the claiming request) when the agent was reported as assigned to an inbound activity.
- URS will detect and remove repeating entries of the same agent into the final list of agents (the one that will be sent to OCS).

ClearTargets

Parameter: `Virtual Queue Name`: STRING or variable (representing a string for the Virtual Queue)

Return value type: VOID

This function removes the current interaction from the virtual queue that is specified as a parameter. When the empty string is specified as a parameter, the current interaction is removed from all virtual queues on which it had been placed.

When selecting a virtual queue, the value of `Virtual Queue Name` can be specified in two ways:

- Enter `Virtual Queue Name` manually.
- From the dialog box, opposite `Virtual Queue Name`, click under `Value`. In the resulting Key dialog box, select a Type (`Virtual Queue` or `Variable`), and then select or enter a value. The values available are for configured virtual queues. The values available for the `Variable` type are the variables included in the Variable list.

Note: The check box `Clear Target` in the Routing Selection object (see Figure 174 on [page 329](#)) frees you from the necessity of manually specifying the `ClearTargets` function in a strategy. This check box automatically generates the same code as the `ClearTargets` function.

If T-Server distributes the `EventDiverted` message with state 22, URS updates `AttributeExtensions` of the `EventDiverted` message with a new `Reason` key, which can contain one of the following values:

- 1 if the call is routed by URS to the target destination.
- 2 if the call is routed by URS to the default destination.
- 3 if the call is routed by the switch to the default destination.
- 4 if the call is cleared from a virtual queue by the execution of strategy function `ClearTargets`
- 5 is used for all other reasons.

For more information, see [page 746](#).

ClearUpdateTrigger

Parameters: `Trigger Key`: STRING or variable (representing a string for the trigger key)

Return value type: none

Use this function to remove the specified trigger key from this list of keys. Using the * character as the trigger key value results in complete cleanup of the list.

CountSkillInGroupEx

Parameters: `Stat Server`: STRING or variable (representing a string for the Stat Server). Uses `default_stat_server` option (see [page 639](#)) if this parameter is evaluated to an empty string.

`Target`: STRING or variable (representing a string for the Agent Group or Place Group, or a comma-separated list of Agents, Agent Groups, Places, or Place Groups)

`Skill Expression`: STRING (variable or constant representing skill expression).

`Sync`: INTEGER (true/false)

Return value type: INTEGER

Use this function to determine the number of agents with a skill set and/or statistical parameters that satisfy the indicated skill expression.

This function returns the number of the agents belonging to the agent group based on the defined skill expression. This function complements the `GetSkillInGroupEx` function ([page 519](#)), which lists the agents in the group.

The `CountSkillInGroupEx` function replaces the `CountSkillInGroup` function. It is similar to `CountSkillInGroup`, but different in that it has a fourth parameter named, `Sync`. It functions in one of two ways:

- If set to true, URS artificially delays execution of the `CountSkillInGroupEx` function if it determines some required statistics to get final results are not opened. The delay is set to 400 milliseconds (ms), after which URS tries to recalculate the content. It repeatedly sets the same delay if there are still some unopened statistics, up to 10 times.
- If set to false, it behaves the same way as the `CountSkillInGroup` function behaved.

IRD hides the old `CountSkillInGroup` function and replaces it with the `CountSkillInGroupEx` function. If a strategy contains the old function, it continues to work like it did before the new function was introduced. URS continues to support it, however, the IRD will always propose the use of the new function in the GUI.

You can also just specify a Stat Server and a skill expression without specifying an Agent Group. However, a skill expression and a Stat Server must be specified. The Stat Server is used to query the content of the provided Agent Group (real or virtual).

When selecting a Stat Server or Agent Group parameter, the value can be specified in the two ways:

- Enter the value manually.
- From the dialog box, click inside the `Value` field and select a Stat Server or Agent Group from the resulting `Key` dialog box. Or click `Variable`, and then select or enter a value. The values available for are the configured parameters. The values available for the `Variable` type are the variables included in the Variable list.

The `StatServer` parameter is the name of the Stat Server containing information on the agents for this function.

The `Agent Group` parameter is the agent group for the Stat Server that this function checks against. Agents are included this group by either placing the agent name from the Persons folder into the Agent Groups folder or defining a virtual group using skill expression within the Annex tab of the Agent Group object. For example, you can create a virtual group for language. In the Annex tab for this virtual group, you create a new section called Language. Within this section, you can create a script option with a value of `Skill[English & Spanish]>3`. (These skills must already have been created in the Skills folder.)

The `Skill Expression` parameter can use skills, variables, numeric constants, and statistics to filter out agents based on their state. The statistic name in a skill expression can be any agent statistic used in the function `SData`. It must be written in the format: `$(statistic)`. This ability to use statistics in a skill expression allows you to conduct queries based on a statistic.

For example, if you want to query the number of agents with a Spanish skill of at least 5 and who are logged in, the expression would be as follows:

```
Spanish>=5 & $(StatAgentsLoggedIn)=1
```

Note: Statistics cannot be used in the Agent Group parameter to filter out agents.

To create this expression in the Skill Expression properties box:

1. Select Skill from the Type column, select Spanish from the Name column, the = operator, a value of 5, and click Add.
2. Next you select the AND button in the same dialog box. Then finally, you select Statistics from the Type column, select StatAgentsLoggedIn from the Name column, the = operator, a value of 1, and click Add again.

The skill expression must be a string constant with enclosing quotation marks; its syntax is described in “Logical Expressions” on [page 92](#).

URS requests the content of any group whether it is real or virtual from Stat Server. If Stat Server is able to provide the information, URS uses the information as the content of the requested group. If Stat Server is unable to provide this information, URS obtains the content of the group from the configuration data. (Stat Server would be unable to provide information if the group does not exist, the skill expression is invalid, or URS cannot connect to Stat Server.)

Unlike the ExpandGroup function, the CountSkillInGroup function does not suspend a strategy for an interaction while waiting for a response from Stat Server. URS uses the content of the group from the configuration data when the function is first called. Once URS obtains the content of this group from Stat Server, that content will be used. URS updates this content any time changes are made that affect this group.

Important Information

- When the name of a configured skill coincides with that of a variable declared in the strategy, the value of the variable is taken into account instead of its homonymous skill. Avoid naming variables after skills that you want to use in the strategy.
- For URS, the statement agent has not the skill SKILL_C equals agent has SKILL_C=0. An agent with SKILL_C=0 is a good target for the condition SKILL_C < 5. In this case, to select agents who have skill SKILL_C but with SKILL_C less than 5, write the following condition: SKILL_C < 5 & SKILL_C > 0 or, alternatively, SKILL_C<5 & SKILL_C != 0.

Warning! The `GetSkillInGroupEx` and `CountSkillInGroupEx` functions do not remove duplicates (if any) from the returned list of Agents/Places. For example, if you use the same Agent Group twice in the Agent Group parameter, the returned list of Agents contains a double set of Agents from this Group.

CountTargetsByThreshold

Parameters:

- `Targets List`: STRING or variable (representing a comma-separated target list)
- `Statistic`: STRING or variable (representing a string for the statistic)
- `Threshold Value`: INTEGER or variable

Note: `Statistic` and `Threshold Value` are rounded to INTEGER before comparison.

`Condition`: `ReadyIfLess` (default), `ReadyIfGreater`, `ReadyIfNotLess`, `ReadyIfNotGreater`

Return value type:

INTEGER (number of targets)

Returns the number of best available target(s) from a list of targets by applying a statistic with a threshold comparison against the input target list. This function returns the number of targets from the `Targets List` parameter (possibly 0).

CreateSkillGroup

Note: Use this function instead of the `GetSkillInGroupEx` function described on [page 511](#). The `CreateSkillGroup` function is dynamic and keeps the connection between the source of the target (agent group or skill expression) and the target itself.

Parameters:

- `StatServer`: STRING or variable (representing a string for the Stat Server)
- `Target`: STRING or variable (representing a string for the Agent Group or Place Group)
- `Skill Expression`: expression or variable

Return value type: STRING. Specification of Agent Group or Place Group with skills.

URS 7.x handles a routing to an agent group using a skill expression exactly like routing to a regular agent group. The only difference is you cannot ask for a statistic on the group of agents that fit the skill expression. URS understands

a target specification like: ?GroupName:SkillExpression@statserver.GA as all agents from <group_name> with the specified skill. This function is a tool to convert the provided Agent Group, Skill Expression, and StatServer into a normal target that represents all agents belonging to the Agent Group supplied as a parameter that satisfies the logical condition given by the Skill Expression. Also see “Important Note on the Absence of a Skill” on [page 366](#).

Note: CreateSkillGroup[StatServer, Agent Group, SkillExpression] is equivalent to Cat['?', Agent Group, ':', Skill Expression, '@', Statserver, '.GA'].

Delay

Parameter: Delay: INTEGER or variable

Return value type: VOID

This function pauses the execution of the strategy for a length of time specified in milliseconds.

ExpandGroup

Warning! The main purpose of the ExpandGroup function in Universal Routing 6.x was to solve priority routing issues. Universal Routing 7.x automatically handles priority routing, there is no need to use the ExpandGroup for this purpose. To do so will result in poor performance. Genesys recommends that you no longer use this function.

Parameter: Target: STRING (variable or constant representing the target)

Return value type: STRING

This function makes it possible to target all Agents (or Places) the Group consists of instead of the Group as a whole. The purpose is to:

- Propagate a target selecting statistic on the Agent level.
- Allow URS to handle the situation where a particular agent is a member of multiple groups and solve related interaction priority issues.

Note: Because Universal Routing 7 can handle priority routing and the most used statistic StatAgentLoading (successor of StatTimeInReadyState) is propagated on the agent level automatically, the value of the ExpandGroup function depreciates in Universal Routing 7.x. Instead of using ExpandGroup, you can return to your original method of specifying targets; for example, target an agent/virtual group or skill expression.

The `ExpandGroup` function returns a comma-separated list of Agents or Place targets belonging to a specified Agent Group or Place Group target in high-level target format.

In IRD, the target types in the dialog box for this function includes:

- `Agent Group` (real and virtual)
- `Place Group`
- `Variable`

For `Agent Group` and `Place Group` targets, you also specify the name of the target for the selected type and its location (Stat Server name). For the `Variable` target type, select the name of the variable only.

This function takes as an argument of a group of Agents or group of Places in one of the following formats:

- `GroupName@Location.GA` (whether a real agent group or virtual agent group)
- `GroupName@Location.GP`
- `GroupName.GA`
- `GroupName.GP`

It returns, respectively, the list of Agents or Places configured in the Group. The list is composed of targets, separated by commas and without spaces, in one of the following formats:

- `AgentName@Location.A`
- `PlaceName@Location.AP`
- `AgentName.A`
- `PlaceName.AP`

`Location`, if present, is the same in the argument and in every target of the output.

This function also supports virtual groups defined using a skill expression in the `Annex` tab of the `Agent Group` object in Configuration Manager.

Note: A group of agents is considered to be *virtual* if agents do not belong to the group permanently; instead, Stat Server assigns an agent to the group when an agent meets the criteria specified for the virtual group. Stat Server adds agents to or removes them from the group if agent parameters that affect eligibility change or if the specified criteria are modified. See the Virtual Agent Groups Chapter in the *Framework 8.1 Stat Server User's Guide* for more information on virtual agent groups.

You can specify a virtual group as the target for the `ExpandGroup` function. URS requests the content of any group whether it is real or virtual from Stat Server.

- If Stat Server is able to provide the information, URS uses the information as the content of the requested group.
- If Stat Server is unable to provide this information about the agent group, URS obtains the content of real agent groups only from the configuration data. For virtual agent groups, URS receives only an empty string since it

is unable to obtain information about agents who fulfill the skill expression from Stat Server. (Stat Server would be unable to provide information if the group does not exist, the skill expression is invalid, or URS cannot connect to Stat Server.)

As a result of URS's request for content from Stat Server, the `ExpandGroup` function—when called the first time—will suspend a strategy for an interaction while waiting for a response from Stat Server. The content of this group will be used. URS updates this content any time changes are made that affect this group.

With this ability to use virtual groups, this function supports skill-based routing.

There is one possible error message:

- `0018 Unknown object`—unable to find the Agents or Place Group in the Configuration Database

ExpandWFActivity

Parameter: `Activity`: STRING or variable (representing a string for the Workforce Management Activity)
`CutOffTime`: Integer

Return value type: String

This function is intended to be used with Genesys Workforce Management. It is similar to `ExpandGroup` (see [page 515](#)) in that it returns a collection of agents as a comma-separated list. It takes as an argument a Workforce Management Activity name and returns the list of agents assigned to the Activity from current moment of time up to next `CutOffTime` number of seconds.

`CutOffTime` is time in seconds that agent has to be assigned to the Activity starting from current moment to be considered as qualified.

Error code 18 is returned if no specified Activity is found.

If an agent assigned to the Activity in the given time interval has a break of any kind (including assignment to another Activity), that agent will not be included in the returned list.

Note: Sometimes this function by design returns an empty list, such as when the `Activity` parameter is not found. When this occurs, the function by design checks the Configuration Database for agents with this `Activity` parameter. If you do not want this to occur, you must configure the strategy to handle the possibility of an empty list being returned.

ExtrouterError

Parameters: Enable (true or false)

Return value type: VOID

Use for external routing to change the default URS reaction in the case of a failure to get a remote access number.

- If the `Enable` parameter is set to `true`, then URS handles an external routing failure as a routing error (according to the `on_route_error` option settings described on [page 648](#)). This prevents URS from ignoring an `EventError` message in response to `RequestGetAccessNumber` and stops URS from continuing to attempt to route based on the original remote access number.
- If set to `false`, URS will continue the attempt to route the call based on the original number. By default, `Enable` is set to `false`.

ExtrouterStatus

Parameter: Switch Name: STRING or variable (representing a string for the switch)

Return value type: INTEGER

This function returns a value of 1 if external routing is possible and a value of <0 if not. The parameter for this function is a remote location where the interaction is being routed.

GetLastErrorInfo

Note: You may wish to use this function after the External Service object. See Table 8 on [page 116](#).

Parameters: none

Return value type: STRING

This function provides detailed information about remote errors. A *remote error* is one reported by any server that URS communicates with. The strategy object that is involved in communicating with the server generates error `0013`, `Remote Error`.

The function should be used immediately after the object that reports the remote error, in order to provide the strategy with detailed information about the error.

The function usually returns value in the `<error_number> <error_message>` format. You need to know numbers of all errors you are interested in.

This function should be directly attached to the red port of this object or to the `0013 Remote Error` branch of an Error Segmentation object (see [page 123](#)).

The most common use of this function is to enable a strategy to handle T-Server's failure to route a call. For example, you might want to have your strategy handle the following failure types differently:

- Destination busy—You might want to try another routing target.
- Invalid ConnID—You might want to terminate the strategy, because the call no longer exists.

Note: In previous releases, any server error was reported as `0013 Remote Error` with no additional information by the `GetLastError` function (retired in 6.x releases) used inside the Error Segmentation object.

This function returns no errors.

Note: Last error information is not stored for long, and can be overwritten by functions that occur subsequently, if they also create errors. To avoid a situation in which `GetLastErrorInfo` reports an incorrect error description, follow the usage guidelines given in this section.

This function can be used to get the http response code in case of a failure. The value is available through the `http_status` key.

GetPriority

Parameters: none

Return value type: INTEGER

This function returns the global priority of the current interaction at the moment when the function is called. Also see the `SelectDN` function on [page 608](#).

GetSkillInGroupEx

Warning! The `GetSkillInGroupEx` and `ExpandGroup` functions, although supported, are deprecated starting in Universal Routing 7. They result in poor performance. Genesys recommends that you not use these functions.

Genesys recommends that you use the `CreateSkillGroup` function (see [page 514](#)) instead of the `GetSkillInGroupEx` function.

`GetSkillInGroupEx` is not dynamic and breaks connection between the source of the target (agent group or skill expression) and the target itself.

The `GetSkillInGroupEx` and `CountSkillInGroupEx` functions do not remove duplicates (if any) from the returned list of Agents/Places. For example, if you use the same Agent Group twice in the Agent

Group parameter, the returned list of Agents contains a double set of Agents from this Group.

Parameters: `StatServer`: STRING or variable (representing a string for a Stat Server)

`Target`: STRING or variable (representing a string for the Agent Group or Place Group, or a comma-separated list of Agents, Agent Groups, Places, or Place Groups)

`Skill Expression`: expression or variable

`Sync`: INTEGER (true/false)

Return value type: STRING

This function returns the list of all the agents belonging to the agent group supplied as a parameter that satisfies the logical condition given by the skill expression. The agents are represented in high-level target format `Name@StatServerName.A`. The different agents in the list are separated by commas. This function complements the `CountSkillInGroupEx` function ([page 511](#)), which provides the number of agents in the group.

This function replaces the `GetSkillInGroup` function. The `GetSkillInGroupEx` function is similar to `GetSkillInGroup`, but different in that it has a fourth parameter named, `Sync`. It functions in one of two ways:

- If set to true, URS artificially delays execution of the `GetSkillInGroupEx` function if it determines some required statistics to get final results are not opened. The delay is set to 400 milliseconds (ms), after which URS tries to recalculate the content. It repeatedly sets the same delay if there are still some unopened statistics, up to 10 times.
- If set to false, it behaves the same way as the `CountSkillInGroup` function behaved.

IRD hides the old `GetSkillInGroup` function and replaces it with the `GetSkillInGroupEx` function. If a strategy contains the old function, it continues to work like it did before the new function was introduced. URS continues to support it, however, the IRD will always propose the use of the new function in the GUI.

You can also just specify a `Stat Server` and a `Skill Expression` without specifying an Agent Group. However, `Skill Expression` and `Stat Server` must be specified.

When selecting a Stat Server or Agent Group, the value can be specified in two ways:

- Enter the string manually.
- From the dialog box, click inside the `Value` field and select a Stat Server or Agent Group from the resulting `Key` dialog box. Or click `Variable`, and then select or enter a value. The values available for are the configured Stat Server or Agent Group. The values available for the Variable type are the variables included in the Variable list.

The `StatServer` parameter is the name of the Stat Server containing information on the agents for this function. `StatServer` is used to query content of provided agent group (real or virtual).

The `Agent Group` parameter is the agent group for the Stat Server that this function checks against. Agent Groups are created through Configuration Manager. Agents are included this group by either placing the agent name from the Persons folder into the Agent Groups folder or defining a virtual group using skill expression within the Annex tab of the Agent Group object. For example, you can create a Virtual Group for language. In the Annex tab for this virtual group, you create a new section called Language. Within this section, you can create an option called, script with a value of `Skill[English & Spanish]>3`. (These skills must already have been created in the Skills folder.)

See *Framework 8.1 Configuration Manager Help* for information on creating Agent Groups.

The `Skill Expression` parameter can use skills, variables, numeric constants, and statistics to filter out agents based on their state. The statistic name in a skill expression can be any agent statistic used in the function SData. It must be written in the format `$<statistic>`. This ability to use statistics in a skill expression allows you to conduct queries based on a statistic. For example, if you want to query the number of agents with a Spanish skill of at least 5 and who are logged in, the expression would be as follows:

```
Spanish>=5 & $(StatAgentsLoggedIn)=1
```

To create this expression in the Skill Expression properties box:

1. Select `Skill` from the Type column, select `Spanish` from the Name column, the `=` operator, a value of 5, and click Add.
2. Next you select the `AND` button in the same dialog box. Then finally, you select `Statistics` from the Type column, select `StatAgentsLoggedIn` from the Name column, the `=` operator, a value of 1, and click Add again.

Skill expression must be a string constant with enclosing quotation marks; its syntax is described in the Logical Expressions section on [page 92](#).

Note: Statistics cannot be used in the `Agent Group` parameter to filter out agents.

URS requests the content of any group whether it is real or virtual from Stat Server. If Stat Server is able to provide the information, URS uses the information as the content of the requested group. If Stat Server is unable to provide this information, URS obtains the content of the group from the configuration data. (Stat Server would be unable to provide information if the group does not exist, the skill expression is invalid, or URS cannot connect to Stat Server.)

Unlike the `ExpandGroup` function, the `GetSkillInGroup` function does not suspend a strategy for an interaction while waiting for a response from Stat Server. URS uses the content of the group from the configuration data when

the function is first called. Once URS obtains the content of this group from Stat Server, that content will be used. URS updates this content any time changes are made that affect this group.

With this ability to use virtual groups, this function supports skill-based routing.

Warning! When the name of a configured skill coincides with that of a variable declared in the strategy, the value of the variable is taken into account instead of its homonymous skill. Avoid naming variables after skills that you want to use in the strategy.

Important Information

- For URS, the statement `agent has not the skill SKILL_C equals agent has Skill_C=0`. An agent with `Skill_C=0` is a good target for the condition `Skill_C < 5`. In this case, to select agents who have skill `Skill_C` but with `Skill_C` less than 5, write the following condition: `Skill_C < 5 & Skill_C > 0` or, alternatively, `Skill_C<5 & Skill_C != 0`.
- Genesys recommends that you not use both the `GetSkillInGroupEx` and `ExpandGroup` functions because they are not dynamic and break the connection between the source of a targets (group or skill expression) and the target itself.

JumpToStrategy

Parameter: Strategy Name: STRING or variable (representing a string for the strategy name)

Return value type: VOID

This function starts a strategy saved as a script in Configuration Layer; the name of the script is passed as a parameter.

Important Information

- When using the `JumpToStrategy` function, be careful that you do not design the new strategy to which the function is calling such that the new strategy calls the original strategy in which the function is located, thus causing an infinite loop.
- There is one possible error message:
`0018 Unknown object—unable to create strategy`

JumpToTenant

Parameters: Strategy Name: STRING or variable (representing a string for the strategy name)

Tenant Name: STRING or variable (representing a string for the name of the tenant)

Return value type: VOID

This function starts a strategy saved as a script in Configuration Layer and belonging to a specified tenant. The names of the tenant and the script are passed as parameters.

Important Information

- There is one possible error message:
0018 Unknown object—unable to create strategy

MultiplyTargets

Parameters: INTEGER (true or false)

Return value type: VOID

This function controls switching into target-multiplication mode. For example, if the value is:

- `false` (default)—URS processes the target selection object in the usual way; When new targets are entered into an existing target selection object, URS waits for the newly-specified targets in this target selection object. The targets that URS waited for before the new targets were entered will be forgotten.
- `true`—When new targets are entered into an existing target selection object, URS creates a new target selection object (not related to the existing target selection object) and adds the new targets into the new target selection object. This change does not affect the existing targets in the existing target selection object).

Note: Do not enter into endless (or extremely long) loops that contain target selection objects when the `MultiplyTargets` function is configured as `true`. Every new entry into an existing target selection object allocates a new internal queue for the call, which can cause the router to run out of memory. By configuring the `MultiplyTargets` function as `false`, you can ensure URS uses a limited number of internal queues.

Sample Use Case

The following sample use case is used to describe how to use the `MultiplyTargets` function. Here are the parameters:

Make a call wait for all agents with `MySkill = 1` with priority 1, with `MySkill = 2` with priority 2, `MySkill = 3` with priority 3, `MySkill = 4` with priority 4, and `MySkill = 5` with priority 5.

Figure 226 on [page 525](#) depicts the old way of entering targets into an existing target selection object, while Figure 227 on [page 526](#) depicts the new way, by using the `MultipleTargets` function.

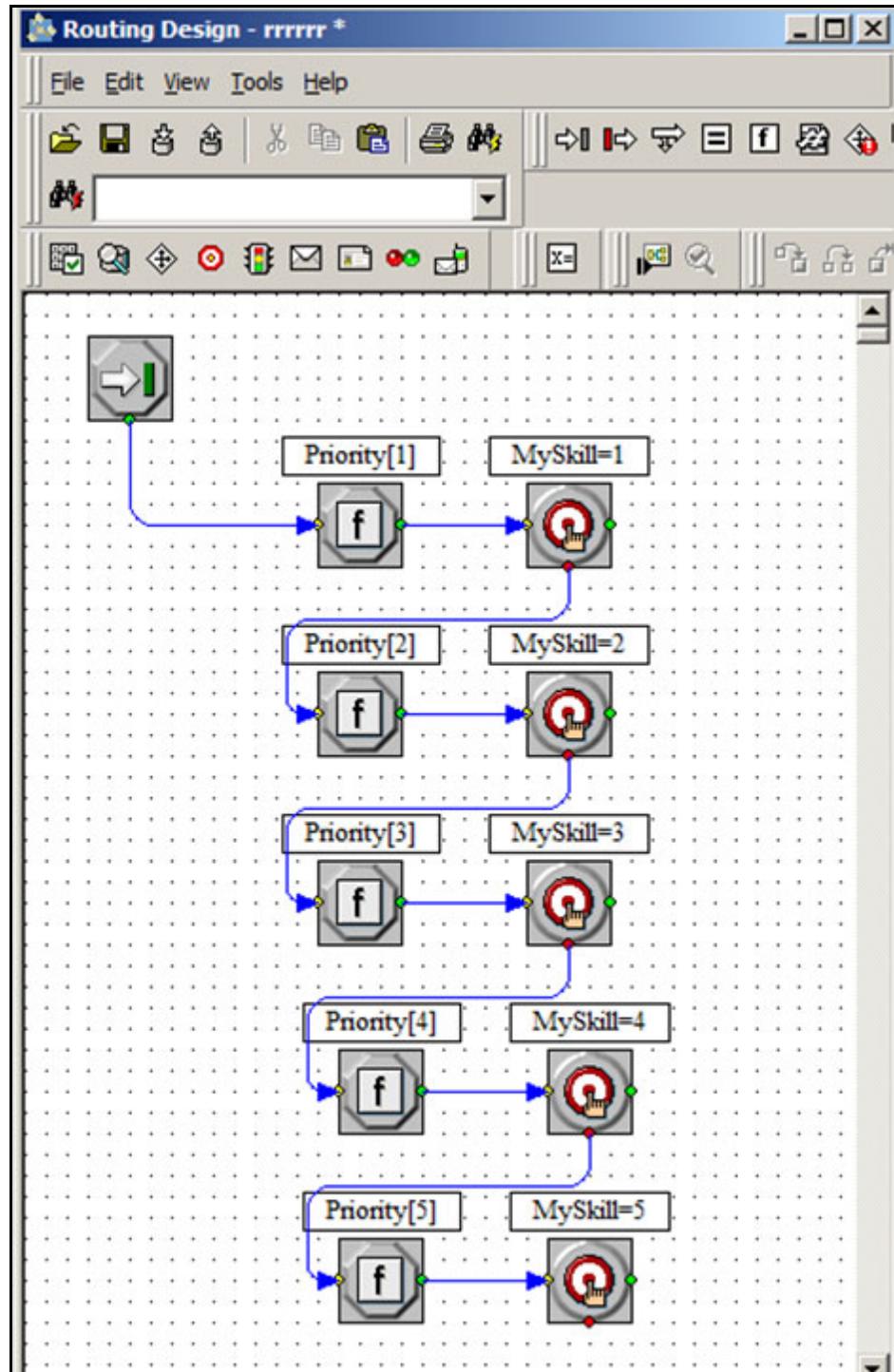


Figure 226: Entering Targets into an Existing Target Selection Object (Old)

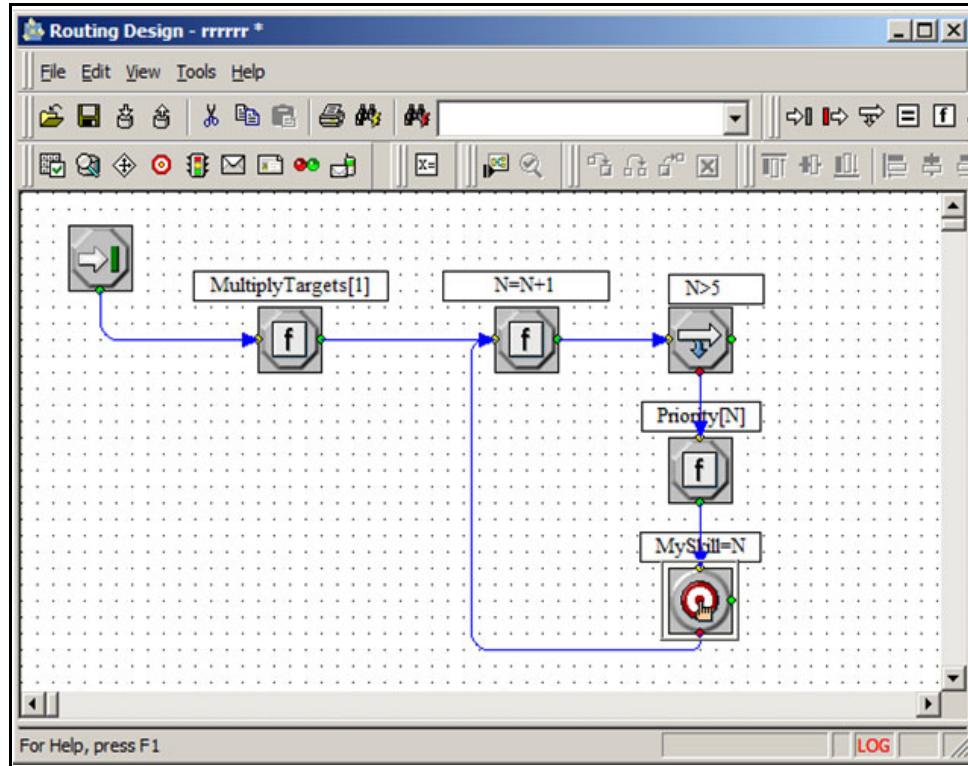


Figure 227: Entering Targets into an Existing Target Selection Object (New)

MultiSkill

Parameters: Stat Server: STRING or variable (representing a string for the Stat Server)

Skill Expression: expression or variable

Return value type: STRING

This function is equivalent to the CreateSkillGroup function with an absent agent group, and returns an agent group in the following format:

? : SkillExpression@0xana_SServer.GA

URS 7.x handles a skill expression exactly like an agent group. The only difference is you cannot ask for a statistic on the group of agents that fit the skill expression. URS understands a target specification like:

?GroupName:SkillExpression@statserver.GA as all agent from <group_name> with the specified skill.

Function MultiSkill no longer returns a list of targets with AGENT type, but an expression as mentioned above.

When selecting a Stat Server, the value can be specified in two ways:

- Enter the Stat Server manually.

- From the dialog box, click inside the Value field and select a Stat Server from the resulting Key dialog box. Or click Variable, and then select or enter a value. The values available are for the configured Stat Server. The values available for the Variable type are the variables included in the Variable list.

The skill expression can use skills, variables, numeric constants, and statistics to filter out agents based on their state. The statistic name in a skill expression can be any agent statistic used in the function SData. It must be written in the format `$(statistic)`. This ability to use statistics in a skill expression allows you to conduct queries based on a statistic.

Skill Expression must be a string constant with enclosing quotation marks; its syntax is described in “Logical Expressions” on [page 92](#).

Important Information

- For URS, the statement `agent has not the skill SKILL_C equals agent has SKILL_C=0`. An agent with `SKILL_C=0` is a good target for the condition `SKILL_C < 5`. In this case, in to select agents who have skill `SKILL_C` but with `SKILL_C` less than 5, write the following condition: `SKILL_C < 5 & SKILL_C > 0` or, alternatively, `SKILL_C<5 & SKILL_C != 0`.
- MultiSkill is equivalent to CreateSkillGroup with an empty group.

OnCallAbandoned

Parameter: Strategy Name

Return value type: VOID

This function specifies the emergency strategy, saved as script in Configuration Layer, to be executed if an interaction is abandoned. The name of the script is passed as a parameter. When the emergency strategy is executed, the interaction no longer exists (it was abandoned). Because the interaction no longer exists, this function is not intended to execute any routing or treatment instructions. The strategy used is intended to provide reporting information; it cannot execute routing instructions.

OnRouteError

Parameters: Error: Integer

Option: STRING or variable

Return value type: VOID

This function allows you to specify an individual URS reaction for every type of error. If used, this option overwrites option `on_route_error` for the current interaction.

In the case of an error, URS behaves in following way:

- Checks if OnRouteError function was executed for this type of error. If yes, then URS behaves as the function specifies.
- Otherwise, checks if there is a URS-hardcoded reaction for this error. If yes, then URS executes this reaction.
- Checks option `on_route_error` (see [page 648](#)) and behaves accordingly.

The following is a list of URS default reactions that are hardcoded (if not overwritten with the `OnRouteError` function) when processing routing errors:

- Regardless of the value that is set for the `on_route_error` option, the following errors are not the result of an unsuccessful target selection, and URS tries to reselect the target (behaves as if `on_route_error` is set to `try_other`):
 - 231 /* DN is busy */
 - 232 /* No answer at DN */
 - 1700 /* Agent reservation attempt failed */
- If the `on_route_error` option is set to `try_other`, and an error is received twice in a row that indicates that the call does not exist (error code: 56 /* Invalid ConnectionID*/), the call is deleted (URS behaves as if `on_route_error` is set to `delete`). The purpose of this is to avoid looping in routing attempts for this scenario.
- If the `on_route_error` option is set to `try_other`, `continue_ok`, or `continue_error` and the error that is received is one of the following: 1214, 1215, 1216, URS repeats the routing attempt. (URS behaves as if `on_route_error` is set to `reroute`).
- If the `on_route_error` option is set to `default`, `reroute`, or `ignore` and a second error message is received, the call is deleted (URS behaves as if `on_route_error` is set to `delete`). The purpose of this is to avoid looping in routing attempts for this scenario.

Print

Note: The Print function in Universal Routing 7.x prints the string in the log at the interaction-level. This means it can show up in the Solution Control Interface (SCI), which allows trouble shooting from SCI.

Parameter: String: STRING or variable (representing a string). The total length should not exceed 16,000 bytes.

Return value type: VOID

This function prints strings in the URS log file. Use a variable if you want to print a series of strings.

Rand

Parameters: Interval: INTEGER or variable

Return value type: INTEGER

This function returns a random integer between a value of 1 and max.

ReleaseCall

Parameters: none

Return value type: STRING (target)

This function releases the current interaction if it has not already been routed.

The returned value can be either `return:ok` or `return:error`.

The function first checks whether the operation is possible. This verification can result in the following errors:

- **0001 Unknown error**—for example, a failure to send request to release interaction.
- **0002 Transfer in progress**—another transfer is in progress; this error message will be returned, if URS has sent an instruction to T-Server to transfer the interaction between two DNs and if, at the time the ReleaseCall function is invoked, no notification has been received by URS about the interaction reaching a DN after the transfer. This state can only be possible because treatments are running while the strategy executes the function.
- **0003 Treatment in progress**—this error will occur if ReleaseCall is invoked after URS has issued an instruction for T-Server to start a treatment but before T-Server responds with EventTreatmentApplied, EventTreatmentNotApplied, or EventError.
- **0004 Call is on hold**—the agent puts the interaction on hold; this error message will be returned either if the interaction is put on hold at a DN for which URS is registered or if the interaction cannot be found on any of the DNs for which URS is registered.
- **0005 Call is gone**.
- **0006 Operation impossible**—the interaction cannot be released now; this error message will be returned if EventRinging has come for the current DN, but EventEstablished has not yet been received.
- **0008 Routing done**—not permitted in postrouting; this message will be returned if the interaction has already been routed successfully by the function RouteCall or by a Routing object, or if the interaction has been force-routed either by the function TRoute or by combining the Force object with a Routing object.

If none of the above errors are discovered, URS instructs T-Server to release the interaction from the current DN.

Note: Release is possible only for a DN that can receive EventEstablished. This function does not work for routing points because EventEstablished is not generated for routing points.

ResetBusyTreatments

Parameters: none

Return value type: VOID

This function clears the buffer of busy treatments and can suspend the strategy for the current call until the call state becomes appropriate for beginning the next treatment.

Usually used by IRD automatically. You can use it explicitly in order to synchronize busy treatments with mandatory treatments—to make sure that a mandatory treatment can be appropriately started.

SelectTargetsByThreshold

Parameters: Targets List: STRING or variable (representing a comma-separated target list)

Statistic: STRING or variable (representing a string for the statistic)

Threshold Value: INTEGER or variable

Note: Statistic and Threshold Value are rounded to INTEGER before comparison.

Condition: ReadyIfLess (default), ReadyIfGreater, ReadyIfNotLess, ReadyIfNotGreater

Return value type:

STRING (comma-separated list of targets)

Finds the best available target(s) from a list of targets by applying a statistic with a threshold comparison against the input target list. This function returns a subset of the Targets List parameter (possibly empty).

Note: There is also a Genesys-supplied macro for postprocessing of output target lists. See “DelimitTargetList Macro” on [page 138](#).

SelectTargets

Parameters: Quota type: QuotaMin, QuotaTarget, QuotaMax, or variable

Targets List: STRING or variable (representing a comma-separated target list)

Return value type: STRING (comma-separated target list)

This function removes from a list of Agent Groups, Place Groups, or queue targets those targets that have already received a number of calls in excess of their quota for the interval when the function is called. These types of quotas are specified in the Configuration Database in `Statistical Days` belonging to `Statistical Tables` of type `Quota Table`, associated with each of the Agent Groups or Place groups. The queue is associated with that `Quota Table` that is associated with an Agent Group for which the queue is an origination DN. If no agent group is found, the queue is associated with the `Quota Table` that is associated to an Agent Group named after the alias of the queue.

The `Quota` type parameter indicates which of the three relevant entries in the statistical day will be treated as the current quota.

The `Targets List` parameter can be:

- A variable for the target list
- A string of comma-separated high-level agent group or place group targets (in the format `Name@StatServerName.GA` or `Name@StatServerName.GP`). Initial and final spaces and spaces contiguous with commas are ignored.

If an Agent Group or Place Group target on the list has no `Quota Table` (see “Configuring Quota Tables Associated to Groups of Agents or Places” on [page 532](#)) associated with it or no `Statistical Day` in the table matches the current date, the target is retained in the list returned by the function: it has not exceeded its quota since no quota was set for it.

Always submit, as the `Targets List` argument, lists that contain only Agent Group and Place Group targets in the complete high-level target format.

However, if the function encounters a list element in a different format, it does one of the following:

- If the element is a syntactically correct target in the complete high-level format but its type is not Group of Agents or Group of Places, the target is retained in the list returned by the function.
- If the element is not a syntactically correct target in the complete high-level format, including targets with an omitted type or location, it will be dropped from the list.

Important Information

- If `SelectTargets` on queues is used, make sure that every queue for which you use `SelectTargets` is listed as an origination DN for at most one Agent Group.
- If more than one group or queue is associated with the same table, then the calls routed to all of them are counted together. That is, the quota is interpreted as a limit on the total number of calls routed to groups and queues associated with the same table. Therefore, you must set up individual `Quota Tables` for groups and queues that you want to consider separately, even if these tables consist of the same `Statistical Days`.

Configuring Quota Tables Associated to Groups of Agents or Places

A Quota Table is configured in Configuration Manager as a Statistical Table object of Quota Table type (see the Configuration Manager documentation for more details on the process of configuration). The Quota Table associated with an Agent Group or a Place Group must be specified inside the Advanced properties of the group. The same Quota Table can be associated with more than one Agent Group or Place Group.

The Quota Table must contain Statistical Days. Use the information about Statistical Days on [page 580](#), but not the information about Statistical Values. The relevant values for Statistical Values are as follows:

Statistical Value 1, Statistical Value 2, and Statistical Value 3 for each Interval of the day—during every interval. Value 1 is used when Quota type has a value of 0 (QuotaMin). Value 2 is used when Quota type has a value of 1 (QuotaTarget). Value 3 is used when Quota type has a value of 2 (QuotaMax).

All other properties of Statistical Days are irrelevant for the purpose of setting up quotas.

Note: The same Statistical Day can belong to more than one Quota Table.

SendEvent

Parameters: Type (type of distributed event):

EventQueued, EventDiverted, EventRinging, EventDialing,
EventEstablished, EventReleased, EventHeld,
EventRetrieved, EventRouteRequest, EventRouteUsed,
EventDigitsCollected, EventAttachedDataChanged,
EventUserEvent, EventAbandoned, EventDialing, EventError,
EventNoEvent, EventPartyAdded, EventPartyChanged,
EventPartyDeleted, EventPrivateInfo,
EventTreatmentApplied, EventTreatmentEnd,
EventTreatmentNotApplied or variable

Event: STRING (list) or variable (representing a string for the event)

Return value type: VOID

URS automatically distributes an event on virtual queues in a strategy. The function provides explicit control to distribute specified events on any desired DN.

The Type parameter is the type of distributed event for this function.

The Event parameter can be empty. If specified, the string or the variable (representing the string) must be in the key-value list format using one of the following keys: event, thisdn, otherdn, thirdpartydn, dnis, ani, collecteddigits, referenceid, userdata, and extensions.

Parameters should appear in key-value pairs. User Data should appear in the format, `userdata.xxxx:yyyy`, where `xxxx` is the key and `yyyy` is the value. For example, to distribute the User Event with attached data `xxxx=12345` and `yyyy = asdfgh`, and `DNIS = 123456789`, then `SendEvent` would be written as follows:

```
SendEvent(EventUserEvent,  
'dnis:123456789|userdata.xxxx:12345|userdata.yyyy:asdfgh')
```

The following error is possible:

- 0001 Unknown error

tserver Key

You can send a User Event to any DN on a T-Server to which URS connects. Do this via the `Event` parameter, which is extended with one more key: `tserver`. For its value, enter the name of the primary T-Server to receive the Event. If no such T-Server exists or is closed, the interaction exits through the red port.

To instruct URS to only send User Events and not react to Events from the T-Server (for example, `EventRouteRequests`), set URS option `event_arrive` to `none` for this T-Server. [Figure 228](#) shows an example of using `tserver`.

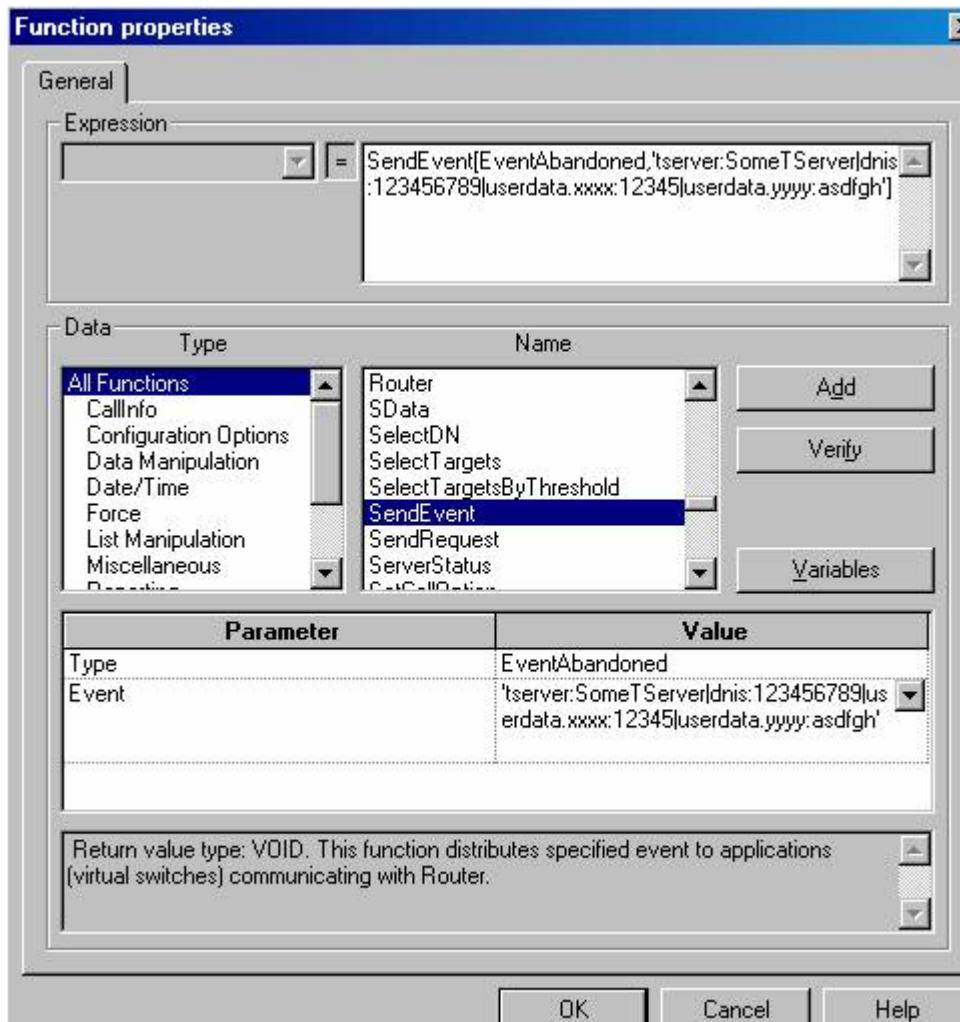


Figure 228: SendEvent Function, Use of tserver Key

SendRequest

Parameters: Type STRING or variable (type of distributed request):
 RequestAgentNotReady, RequestAgentReady,
 RequestAnswerCall, RequestApplyTreatment,
 RequestCallForwardSet, RequestDistributeUserEvent,
 RequestMakeCall, RequestPrivateService,
 RequestRedirectCall (see note below), RequestReleaseCall,
 RequestRouteCall, RequestUpdateUserData
 Request: STRING (list) or variable (representing a string for the request)

Return value type: Integer Reference ID of sent request

This function, used by several of the predefined macros (see “Macro” on [page 130](#)) makes it possible to send Request messages to T-Server, such as messages notifying of ring-no-answer situations (see [Figure 229](#)).

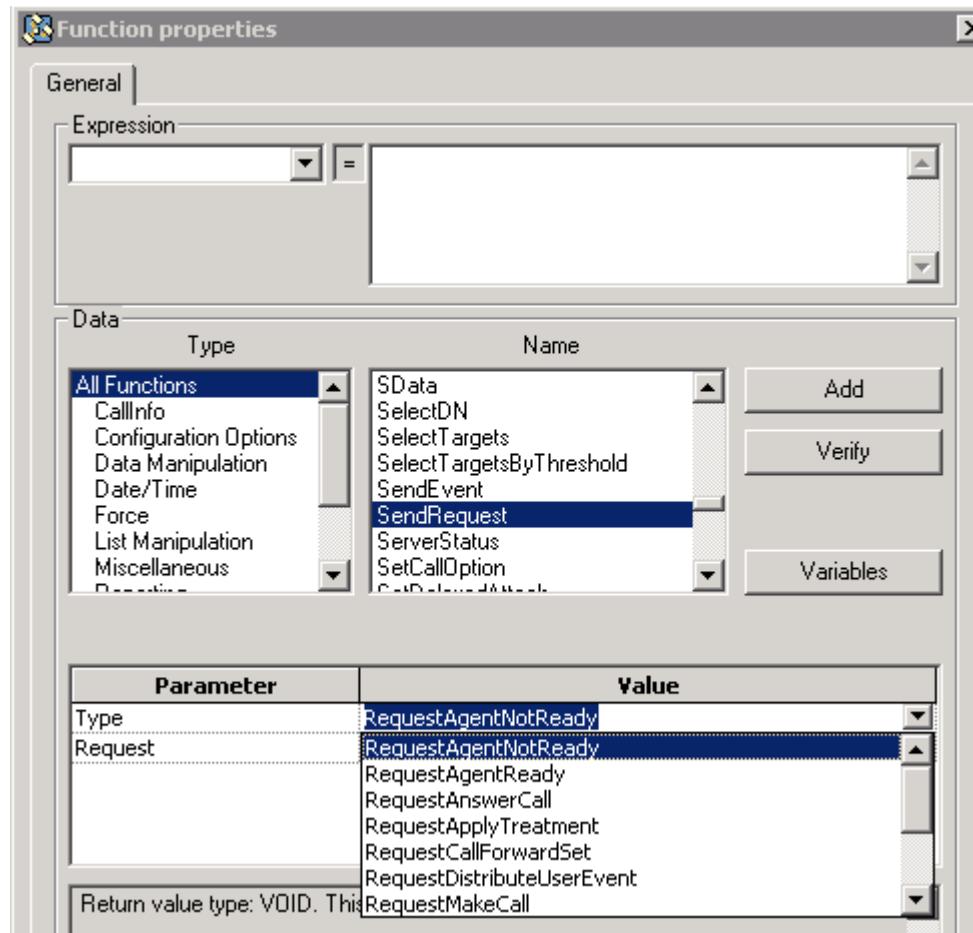


Figure 229: SendRequest Function, Value Dropdown Menu

The Type parameter is the type of request for this function.

Note: You must use the Exit object (see [page 127](#)) in the green port branch of RequestRedirectCall. If either the RedirectCall or RedirectCallMakeNotReady macro (see [page 130](#)) is used within a strategy, you must end the green port branch of those objects with the Exit object. The red port is used only if the last function in the macro returns an error; otherwise the interaction goes through the green port.

To determine what event and request messages your T-Server supports, consult your particular T-Server's deployment guide. For example, if your T-Server is Avaya ECS (MV), consult the *Framework 8.1 T-Server and HA Proxy for Avaya DEFINITY ECS (MV) Deployment Guide*.

Each T-Server deployment guide has a “Supported Functionality” section (Part Two). This section contains a table with the following columns: Feature

Request, Request Subtype, Corresponding Event, Supported. [Table 134](#) shows an example.

Table 134: Example T-Server Supported Functionality

Feature Request	Request Subtype	Corresponding Event	Supported
TCallSetForward	None	EventForwardSet	Y
	Unconditional		Y
	OnBusy		Y
	OnNoAnswer		Y
	OnBusyAndNoAnswer		Y
	SendAllCalls		Y

The Request parameter in SendRequest can be empty. If specified, the string or the variable (representing the string) must be in the key-value list format using one of the following keys: event, thisdn, otherdn, thirdpartydn, dnis, ani, collecteddigits, referenceid, userdata, and extensions.

The SendRequest function supports an additional parameter that can be specified in the body (the Request parameter value) of the request: tserver. It enables URS to send requests on any T-Server to which URS is connected (not just T-Servers that host currently processed interaction). This parameter specifies the name of a T-Server in the URS connection list and the request is sent to this T-Server or its backup, whichever one is alive.

Handling Ring-No-Answer Situations

If your switch/T-Server supports it, the SendRequest function can be used to handle ring-no-answer situations at the strategy level. An example ring-no-answer scenario is as follows:

- An agent steps out without selecting the desktop function that assigns an agent not ready state.
- A call comes in, which URS distributes to the agent who has stepped out.
- Since the agent is away, the call rings but is not answered. Instead, the call distributed by URS to the agent is lost.

To prevent this type of situation from occurring, URS must:

- Determine that a ring-no-answer situation exists.
- Redirect the call to another available agent without wasting any additional wait time.

- Prevent further distribution of calls to the agent until he/she becomes available again.

Implementation

IRD provides and URS implements a ring-no-answer solution using function SendRequest, which makes it possible to send to send various requests (see the dropdown menu in Figure 229 on [page 535](#)) to T-Server. The ring-no-answer solution presented here requires only two of the available requests presented in the SendRequest dropdown menu Figure 229 on [page 535](#): RequestAgentNotReady and RequestRedirectCall.

The redirect functionality is implemented on the strategy level as follows:

- URS is registered on all agents DNs.
- An initial strategy loaded on these DNs checks for a ring-no-answer situation.
- If the ring-no-answer situation exists, a second strategy redirects the call to some local routing point.
- At the local routing point, a third strategy routes the call to an available agent.

Note: In general, only local redirection is possible, so the third strategy is required.

The IRD Macro object (see [page 130](#)) provides the interface for implementing the solution with three predefined macros:

1. RedirectCall[Local Routing Point]
2. RedirectCallMakeNotReady[Local Routing Point]
3. MakeAgentNotReady[]

Determining Whether Your T-Server Supports Redirect

Since the possibility of issuing RequestRedirectCall from a routing strategy is a basic part of the ring-no-answer solution at the strategy level, you must first determine whether your T-Server supports this type of request.

For the purpose of making this determination, Genesys T-Servers are grouped into three different groups:

- Group #1: T-Servers that support the TRedirect command based on switch functionality (example: T-Server NEC NEAX/APEX).
- Group #2: T-Servers that can emulate the TRedirect command when the switch does not support this functionality (example: T-Server EADS Intecom M6880).
- Group #3: T-Servers that do not support the TRedirect command (example: T-Server Meridian).

T-Server Groups #1 and #2

T-Server Groups #1 and #2 are identical since they provide identical TEvent flow to their clients. Both kinds of T-Servers fully support implementation of the ring-no-answer solution presented here without restrictions. [Figure 230](#) shows an example simple redirecting strategy for the solution:

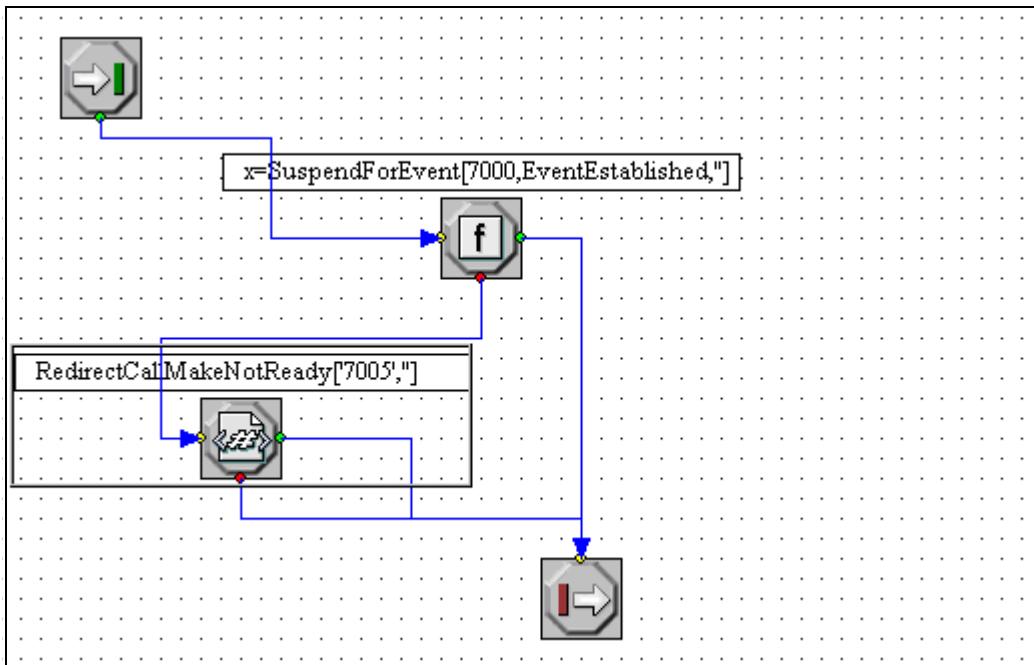


Figure 230: Example Simple Redirect Strategy

This type of redirecting strategy is loaded on every agent's DN. The signal for URS to start the strategy is EventRinging on a DN.

Loading the Redirecting Strategy on Agent DNs

1. In Configuration Manager, highlight all the desired DNs.
2. Right-click to bring up a context menu.
3. Right-click and select **Manage Options** from the shortcut menu. The Multiple Objects Update Wizard opens (See [Figure 231](#)).

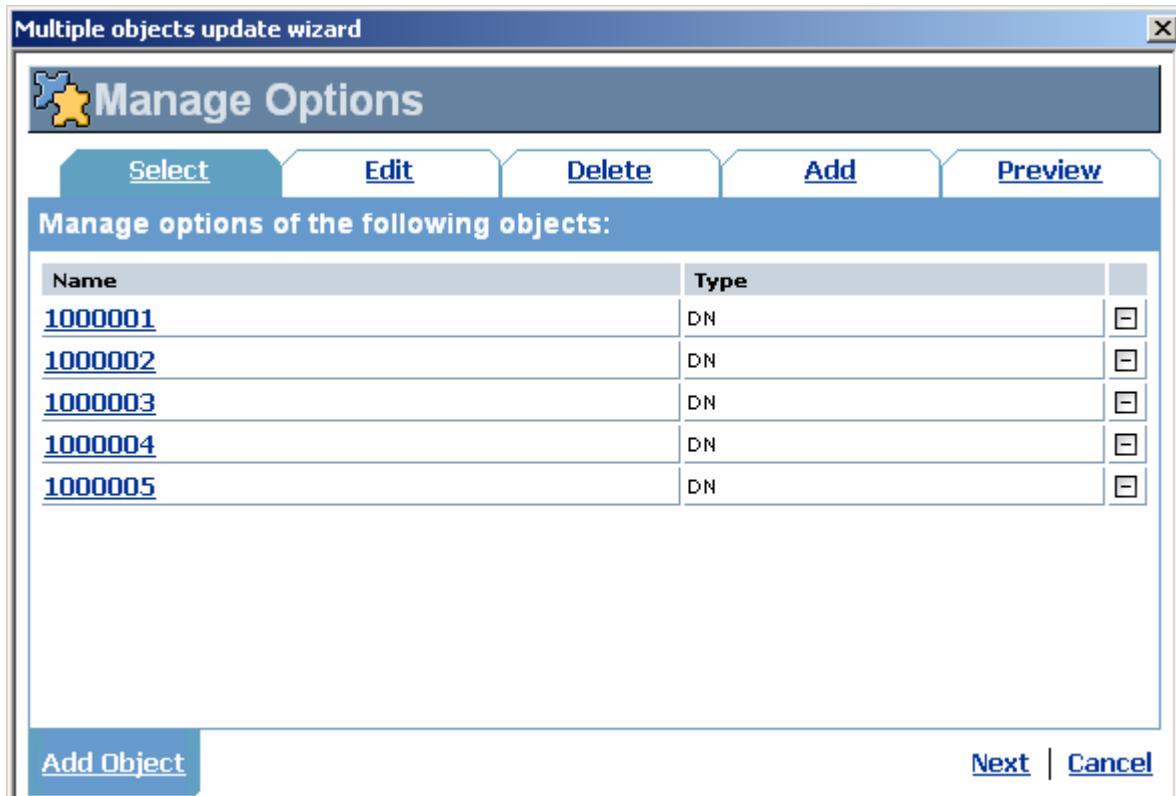


Figure 231: Configuration Manager Multiple Objects Update Wizard

This wizard takes multiple selected objects and allows you to simultaneously apply identical operations to options in the Annex tab.

4. Click Next and then Modify Annex in the Multiple Objects Update Wizard and do the following into the Annex tab of each DN:
 - Create a section with the name of the URS Application.
 - Create option strategy. Assign the value as the name of the redirecting strategy.
 - Create option event_arrive with the value ringing
5. If a backup URS is used, add the same options for it as well.

What Happens During Ring-No-Answer

With the above solution, the following happens:

- After EventRinging happens on the agent DN, URS starts the strategy (see Figure 230 on [page 538](#)). The first operand in the strategy is the SuspendForEvent function for EventEstablished on this DN (EventEstablished is a sign that a call was answered by an agent).
- If there is no EventEstablished after a specified timeout (7 seconds in this case), then macro RedirectCallMakeNotReady is executed. This macro sends two requests to T-Server: RequestRedirectCall to a local route point (7005) and RequestAgentNotReady.

- As a result, the call is redirected to the local route point and the agent sent into a not ready state.
- The strategy loaded on the local route point redistributes the call to another available agent.

To keep the call's original position in queue and avoid additional waiting for redirected calls, you could use the PriorityTuning function (see [page 603](#)) with UseAgeOfInteraction parameter set to true. In this case:

- Functions PriorityTuning (UseAgeOfInteraction = true) and SetInteractionAge (keep = true) are added to the main routing strategy.
- Function PriorityTuning (UseAgeOfInteraction = true) is added to the third strategy to switch on using Age Of Interaction priority tuning on the local route point.

T-Server Group #3

If your T-Server belongs to Group #3 (see [page 537](#)), there is no possibility of redirecting an existing call to another available agent when a ring-no-answer situation occurs. However, it is still possible to put the agent into a not ready state. [Figure 232](#) uses the predefined macro MakeAgentNotReady.

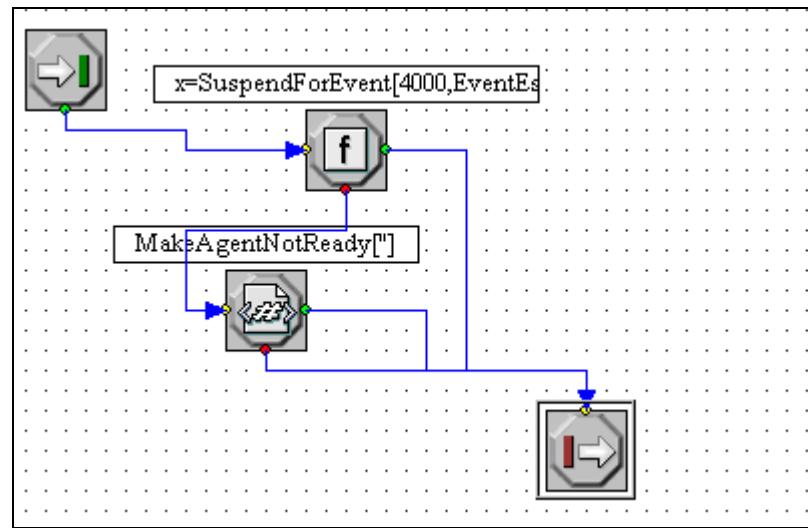


Figure 232: Example Simple MakeAgentNotReady Strategy

In the strategy in [Figure 232](#), after the waiting time for EventEstablished expires, macro MakeAgentNotReady sends RequestAgentNotReady. The call that activated this strategy is lost, but the agent is sent into a not ready state.

Note: If loss of the first call is not acceptable, then the call can be delivered to a local route point by transfer (for example, TRoute function). In this case, the call will be counted as answered by Stat Server.

Custom Macro Examples

In many cases, the predefined macros (see [page 74](#)) will satisfy the requirements for a ring-no-answer solution. In other cases, the predefined macros can serve as samples for creating custom macros to meet specific customer requirements.

Example #1

The predefined macro RedirectCallMakeNotReady issues a RequestAgentNotReady where attribute ThisDN is the same as ThisDN in the event that started strategy (in this case, the number of the DN where the strategy is loaded). For some T-Servers, this may not be acceptable.

Example #2

On the Meridian switch, an agent usually has two DNs: Extension and Position. A call from a route point can be distributed only to the agent's Extension DN, but in macro RequestAgentNotReady, T-Server expects the number of the Position DN. In this case, you would need to define a custom macro named, for example, macro MakeAgentNotReady_Meridian, which has two input parameters: thisdn and queue (see [Figure 233](#)).

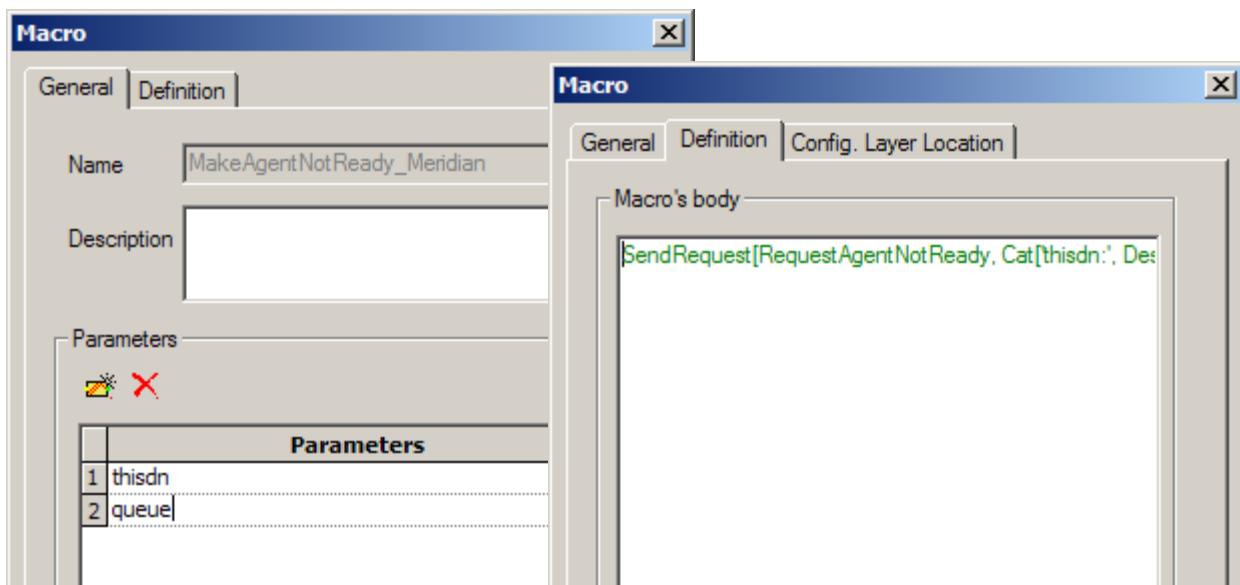


Figure 233: Example Custom Macro, MakeAgentNotReady_Meridian

It is assumed that number of the agent's position DN is defined in the strategy and sent to the macro. [Figure 234](#) shows strategy which is loaded on the agent Extension DN and able to determine number of the agent's position DN.

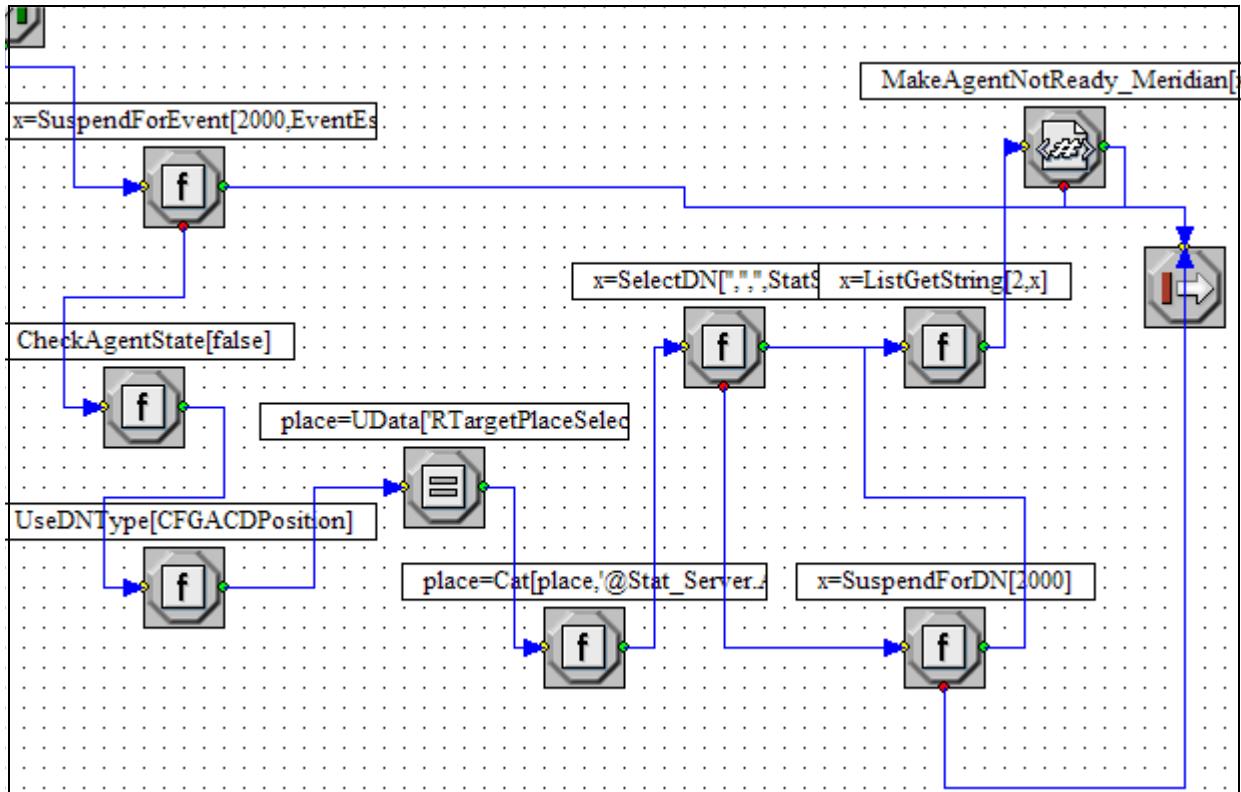


Figure 234: Example Strategy for Obtaining Agent's Position DN

In the strategy in Figure 234:

- The URS option report_targets is set to true in order to have key-value pair RTTargetPlaceSelected (agent's Place attached to the call).
 - The sequence of functions is as follows:

```
CheckAgentState[false]
UseDNTType[CFGACDPosition]
place = UData[RTargetPlaceSelected]
place = Cat[place, '@Stat_Server.AP']
x = SelectDN['', '', '', '', place, '', '', '', '', '', '', '', '', '']
x = SuspendForDN[2000]
x = ListGetString[2, x]
```

where place and x are variables invoking the agent's position DN.

- The macro `MakeAgentNotReady` `Meridian` is called.

Example #3

Predefined macro `RedirectCallMakeNotReady` issues requests in the following sequence:

1. RequestAgentNotReady
 2. RequestRedirectCall.

The G3 Switch (T-Server Avaya Definity ECS (MV)) cannot process RequestAgentNotReady while EventRinging happens on agent's DN. In this case, you could create a custom macro, which sends RequestRedirectCall first.

T-Servers That Do Not Provide TRedirect

As alternative for T-Servers that do not provide TRedirect in any form, but support SetCallForward, this type request might be used to address ring-no-answer scenarios. Here, the agent needs to recognize and cancel forwarding when he/she becomes available again.

In this case:

- Further distribution of calls to the agent cannot be prevented (until he/she becomes available again).
- All calls (except for the first one that activates SetCallForward) are forwarded to an additional local route point.
- An additional routing strategy routes calls back to another local route point where a strategy for redistributing calls is loaded.

This type of ring-no-answer solution requires four strategies:

1. The main routing strategy.
2. A forwarding strategy loaded on agent's DN.
3. An additional strategy loaded on additional local route point.
4. A redistributing strategy loaded on a local route point.

The additional local route point is necessary due to the impossibility of forwarding a call to the route point from which it arrived. So if a call was redistributed by a strategy from route point A to the agent, it should be forwarded first to route point B and then routed back to route point A.

[Figure 235](#) shows an example forwarding strategy.

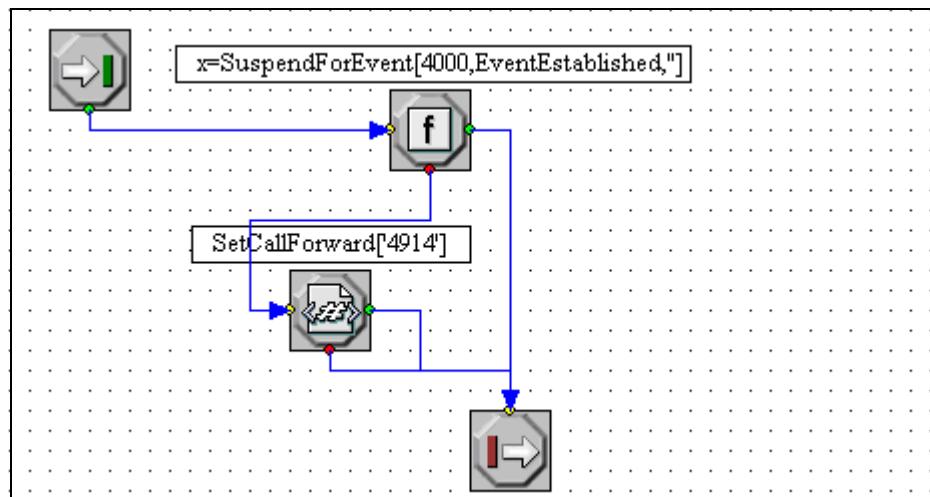


Figure 235: Example Forwarding Strategy

Figure 236 shows the macro SetCallForward used in the example forwarding strategy.

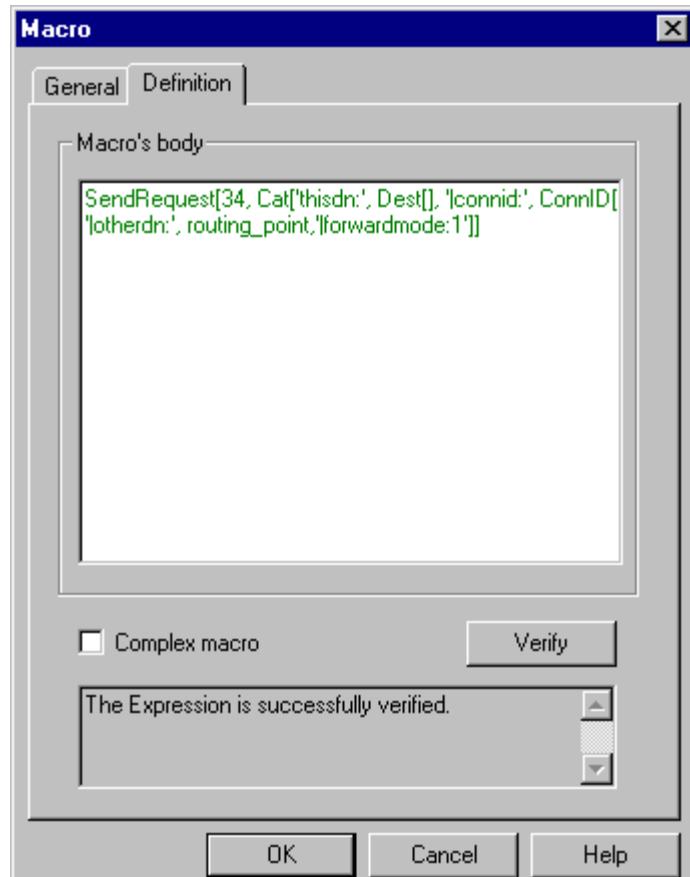


Figure 236: Example Custom Macro CallSetForward

ServerStatus

Parameter: Server name: STRING or variable (representing a string for the server)

Return value type: INTEGER

This function verifies the status of the specified server. It returns a value of -2, -1, or 1. A return of -1 means that the server is not available; a 1 means that the server is available. For T-Servers, a return of -2 means URS is connected to the T-Server but the T-Server is not connected to its switch; 1 means URS is connected to the specified T-Server and the T-Server is connected to its switch.

When selecting a Server, the value can be specified in two ways:

- Enter the Server name manually.

- From the dialog box, click inside the `Value` field and select a Stat Server from the resulting `Key` dialog box. Or click `Variable`, and then select or enter a value. The values available for are the configured Server. The values available for the Variable type are the variables included in the Variable list.

SetDelayedAttach

Parameter: Delay Flag: INTEGER (1 for true or 0 for false)

Return value type: VOID

The `SetDelayedAttach` functionality was previously used implicitly inside the `MultiAttach` object, but is now exposed as a Function. The valid values are 1 (true) and 0 (false). All invocations of the `Attach` or `Update` functions made after the occurrence of a `SetDelayedAttach` function set to `true` now propagate update events to the T-Server only after a `SetDelayedAttach` function set to `false` is called, or strategy execution for this interaction is suspended for some reason.

SetInteractionAge

Note: For additional information, see the section on business-priority routing in the *Universal Routing 8.0 Routing Application Configuration Guide*.

Parameters: Keep: INTEGER (true or false)

Return value type: VOID

Use this function to overwrite the default age of interaction. `SetInteractionAge` can be useful if the age of the interaction will be used for placing interactions into waiting queues (see the `PriorityTuning` function on [page 603](#)).

Definition: Interaction age is the time accumulated since the interaction was known by Genesys (normally set at the very first route point the interaction enters).

By default, interaction age is defined by the time of the last `EventRouteRequest`. However, if an interaction is going to be routed more than once (for example, if an agent transfers an interaction on a routing point for re-routing or if the Voice Callback Universal Callback Server resubmits a callback interaction to URS), the time of the last `EventRouteRequest` is not always the best way to define interaction age.

- `SetInteractionAge[true]` fixes the current interaction age so that it does not depend on subsequent routing events.
- `SetInteractionAge[false]` unfixes the age of interaction so it again will be defined by the moment of the last `EventRouteRequest`.

Setting up Priority Tuning with Age of Interaction

Use function SetInteractionAge(Keep) to timestamp an interaction. Age of interaction will be counted from the timestamp moment on.

Then use function PriorityTuning (UseAgeOfInteraction, <...>) to prioritize this interaction among others in existing queue according to their age. Set parameter UseAgeOfInteraction to true

Use Case:

- Interactions queued by FIFO.
- Business process and call flow requires many transfers/collaborations.
- The age of interaction should be known in order to adjust priority based on age.

In a strategy, the setup might appear as shown in [Figure 237](#).

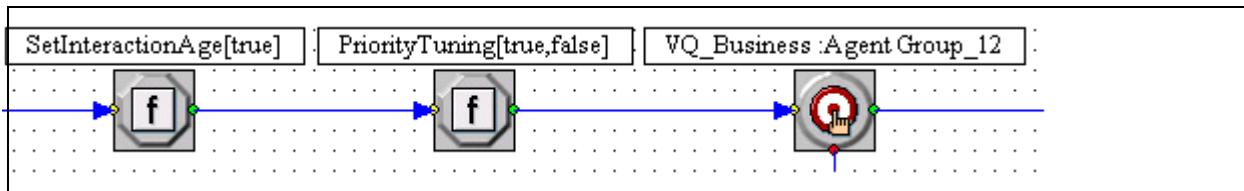


Figure 237: Example of SetInteractionAge Function in Strategy

Note: Once set with SetInteractionAge[true], the age of interaction remains in effect (persist with interaction) for the rest of interaction life or until explicitly cleared with SetInteractionAge[false].

SetLastError

Parameters: Error Code (INTEGER)

Return value type: VOID

Usually error conditions in strategies are raised by some error that occurs while the interaction is handled by a strategy object. This function enables the strategy to generate an error based on a strategy decision.

For example, after analyzing subroutine input parameters, this function can generate an error if one or more parameters are invalid. Use it to flag subroutine errors by specifying what is considered an error in a subroutine (beyond just flowing out the red error port). Then, when control is returned to the calling strategy, the flow continues out the red port of the Call Strategy object.

For example, assume subroutine F[a, b] should return a/b. Then inside the strategy, you can check if b is 0 and, if it is, then set an error code and return.

The interaction flow continues through the red port of the Call Subroutine object.

The input parameter of this function is an error code (for the list of error codes, see Table 12 on [page 125](#)). **Figure 238** shows the error code dropdown list in the properties dialog box for the SetLastError function.

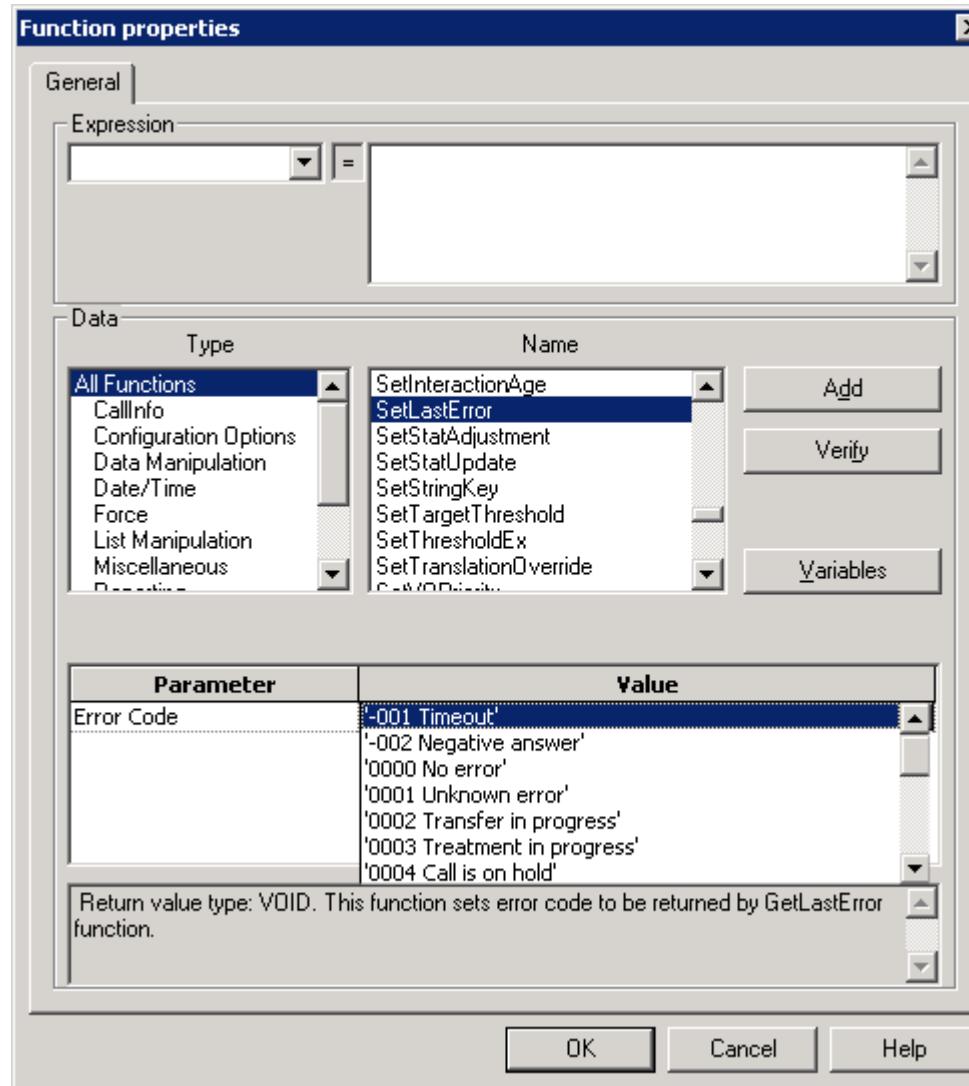


Figure 238: SetLastError Function

Note: Do not be confused by the text that follows each error code, which is there simply for the convenience of the user. URS works with integer error codes. Because of the explanatory text that follows, IRD uses strings, but these strings are converted to real numbers in order for URS to work with them. The starting part of the string is converted to a number; everything else is discarded.

Control will flow out the red port by selecting any item in the list except '0000 No error.'

Error Codes That Makes Sense

Assume you have a strategy that performs a database lookup and the lookup returns null. Technically, the null is not an error, but, in terms of strategy/business objectives, it could be an error. If there is a condition checking for this, the error branch could go right to a SetLastError function object.

In the SetLastError function object, if you pick any item in the dropdown list except for '0000 no error' (see Table 12 on [page 125](#)), control would go out the red port to a Call Subroutine object in the calling strategy. Though this might be the desired result, it is better to pick something from the list that makes sense for the given error condition. For the above example, '0022 No data' is a logical choice. If you picked '0012 Bad target', this selection would work, but could be confusing for another developer studying your strategy.

If you want to flag a branch in a strategy as an error by using a variable, it is recommended to use an INTEGER variable with a non-zero value. This variable can then be passed as the input parameter to the SetLastError function. Pick a non-zero value that maps to one of the strings in the dropdown that makes sense given the error condition.

Note: Also see the object “Error Segmentation” on [page 123](#).

SetQueueLabel

Parameters: Label: STRING

Return value type: VOID

This function is used to label interactions placed into an internal routing queue which represents the internal URS structure for queuing interactions waiting for the same targets.

The Label value set by the SetQueueLabel function is used as Label parameter in the RvqData function, [page 570](#), to retrieve the specific metric.

SetRunTimeMode

Parameters: Flags RunTimeModeManualVqDivert, RunTimeModeStrictAbandon, RunTimeModeNoSubLists

Return value type: INTEGER

The `SetRunTimeMode` function allows the user to specify the run time rules for a specific call. It accepts the same input parameters as the `run_time_mode` option, or their symbolic values.

Select the `Flags` parameter value from the drop-down menu, or enter the corresponding numeric value manually.

Supported symbolic and numeric values:

- `RunTimeModeManualDivert` (numeric value 16). Disables the automatic distribution of `EventDiverted` to Virtual Queues when a call is successfully routed. `EventDiverted` will be postponed until its distribution is explicitly requested in the strategy by the `ClearTargets` function, or the call is deleted from URS memory.
- `RunTimeModeStrictAbandon` (numeric value 64). Disables the target selection, defined in the strategy as objects or functions, immediately after `EventReleased` or `EventAbandoned` is received.
- `RunTimeModeNoSubLists` (numeric value 128). Disables support of sublists while operating with List Objects. As a result, the dot (.) character is considered as a regular character within a key name.

SetUpdateTrigger

Parameters: `Trigger Key`: STRING or variable (representing a string for the trigger key).

`Reaction`: INTEGER

Return value type: VOID

Use this function to add or overwrite a new trigger key in a list of keys.

For every interaction, URS maintains a list of attached data keys that it will check on any `EventAttachedData` event. If the value of any of these keys changes, URS additionally performs special actions on this interaction. To manipulate this list of stored keys URS/IRD provide this `SetUpdateTrigger` function, as well as `ClearUpdateTrigger` (see “`ClearUpdateTrigger`” on [page 511](#).)

There are currently two valid reactions that can be set for the trigger key:

`UPDATE_TRIGGER_SETPRIORITY` (integer value 1)

`UPDATE_TRIGGER_TARGETEXIT` (integer value 2).

URS works with triggers in the following way:

- If attached data was changed for at least one of the trigger keys that have reaction `UPDATE_TRIGGER_SETPRIORITY`, URS takes the current values for all such keys, interprets them as numeric values, and selects the maximum value among all keys with reaction `UPDATE_TRIGGER_SETPRIORITY`. URS changes the priority of the call in all its internal queues to this value, and

also sets the call default priority to the same value so it will be used as the default priority when (if) the call is placed in new queues. So, if the new priority value is N, then URS performs the following functions:

`Priority[N]` and `SetVQPriority['', N]`.

- If attached data was changed for at least one of the trigger keys that have reaction `UPDATE_TRIGGER_TARGETEXIT`, URS checks for current calls that are currently waiting for ready targets (that are located in a Routing Rule or `SuspendForDN` function) and if found it will immediately quit from them through red port and continue the strategy. The raised error code in this case will be `-003 Exit`. This is identical to the functionality that URS executes upon an `Exit busy` treatment.

Both types of triggers are checked independently and it is possible that both of them will be activated.

StrFormatTime

Parameters: `Format: STRING`

`Time: INTEGER`

`Local: INTEGER (true or false)`

Return value type: `STRING`

This function returns Time formatted as string based on the input parameters:

`Format` is identical to the CRT function `strftime` (<http://www.cplusplus.com/reference/clibrary/ctime/strftime/>).

`Time` is UTC timestamp.

`Local` is timezone selection flag:

- If the flag is set to `true`, the timestamp will be interpreted in local time zone.
- If the flag is set to `false`, the timestamp will be interpreted in GMT time zone.

SuspendForEvent

Parameters: `Timeout: INTEGER or variable`

`Type: (type of distributed event): EventQueued, EventDiverted, EventRinging, EventDialing, EventEstablished, EventReleased, EventHeld, EventRetrieved, EventRouteRequest, EventRouteUsed, EventPartyChanged, EventDigitsCollected, EventAttachedDataChanged, EventPartyDeleted, or EventUserEvent or variable.`

`Event: STRING (list) or variable (representing a string for the event)`

Return value type: `STRING`

This function synchronizes a strategy's execution with an external event. URS will suspend a strategy's execution until it receives the specified event. Upon receiving the event, the function returns event_description in a key-value list format as specified in the SendEvent description with an additional key, return:ok. If the event doesn't arrive within the specified timeout, the function returns the string, return:timeout.

The Timeout parameter is the maximum time, in milliseconds, to wait for the event.

The Type parameter is the type of distributed event for this function.

The Event parameter can be empty. If specified, the string or the variable (representing the string) must be in the key-value list format using one of the following keys: event, thisdn, otherdn, thirdpartydn, dnis, ani, collecteddigits, referenceid, userdata, and extensions.

The SuspendForEvent function can be used for the EventPartyChanged event when the AttributeConnID and AttributePreviousConnID event attributes are identical (and the call associated with this ConnID is running the strategy).

URS also supports the use of the SuspendForEvent function for EventPartyChanged event when the consult call process starts. When processing starts, the SuspendForEvent function is invoked (for the EventPartyChanged event) and the EventPartyChanged event, which signals transfer completion, arrives.

Parameters should appear in key-value pairs. User Data should appear in the format, userdata.xxxx:yyyy, where xxxx is the key and yyyy is the value. For example, a string value to suspend a strategy for no more than 5 seconds (5000 milliseconds), until the data xxxx=12345 is attached, SuspendForEvent would be written as follows:

```
SuspendForEvent(5000, EventAttachedDataChanged,  
'userdata.xxxx:12345')
```

The following error can possibly occur: -001 Timeout

Note: The SuspendForEvent function can only correctly detect events that contain the ConnectionId attribute.

TargetComponentSelected

Parameters: none

Return value type: STRING

This function returns the agent-level target to which the interaction was routed definitively.

- If the interaction has not been routed definitively yet, the function returns an empty string.
- If the target specified in strategy and selected for routing is of type Agent, Place, Queue, or Routing Point, the function returns this target itself.

- If the target type is Agent Group, Place Group, Queue Group the function returns the agent, place or queue from the corresponding group the interaction was sent to.

TargetObjectSelected

Parameters: none

Return value type: STRING

This function returns the high-level target (one that you specify in a strategy) to which the interaction was routed definitively. If the interaction has not yet been routed definitively, the function returns the empty string. If a skill expression is used, the function returns:

? : SkillExpression@statserver.GA

or even

?GroupName : SkillExpression@statserver.GA

See “Function Comparison” on [page 552](#).

TargetSelected

Parameters: none

Return value type: STRING

This function returns the DN and the switch name of the target to which the interaction was routed definitively; the result is returned in an apparent high-level target format (Name@StatServerName.Type). If the interaction has not yet been routed definitively, the function returns the empty string. The Type of the target selected is one of the following high level Types as described in the *Framework 8.1 Stat Server User's Guide*: Agent, Place, Queue, QueueGroup, RoutePoint, GroupAgents, or GroupPlaces.

Function Comparison

Every routing destination can be described on three levels: Group, Agent, and DN. Each level has its own function.

- TargetObjectSelected for Group level.
- TargetSelected for DN level
- TargetComponentSelected for Agent level

If the Group level is absent (for targets of type Agent or Place, for example), the functions TargetObjectSelected and TargetComponentSelected are identical. Since URS considers a skill expression as a group of agents, all three functions can be used to report skill criteria.

Example Output

Assume the following:

- The Agent is ready on a Switch DN and is member of an Agent Group.
- The Agent is associated with a Place and the Place is member of a Place Group.
- The Agent has the skill Spanish = 5.
- A DN of type ACDQueue with number 2222 on the Switch, alias Queue2222, that is member of a Queue Group.

[Table 135](#) shows the results of using each function for different targets.

Table 135: Target Function Comparison

Target:	Function TargetSelected	Function TargetObjectSelected	Function TargetComponent Selected
Agent@statserver.A	DN@Switch.A	Agent@statserver.A	Agent@statserver.A
Place@statserver.AP	DN@Switch.AP	Place@statserver.AP	Place@statserver.AP
AgentGroup@statserver.GA	DN@Switch.GA	AgentGroup@statserver.GA	Agent@statserver.A
PlaceGroup@statserver.GP	DN@Switch.GP	PlaceGroup@statserver.GP	Place@statserver.AP
MultiSkill(Spanish>3)	DN@Switch.GA	?:Spanish>3@statserver.GA	Agent@statserver.A
CreateSkillGroup(Agent Group, Spanish>3)	DN@Switch.GA	?AgentGroup:Spanish>3@statserver.GA	Agent@statserver.A
Queue222@statserver.Q	2222@Switch.Q	Queue222@statserver.Q	Queue222@statserver.Q
QueueGroup@statserver.GQ	2222@Switch.GQ	QueueGroup@statserver.GQ	Queue222@statserver.Q

Note: The `report_targets` option controls whether URS attaches information on high level routing targets to the interaction. See [page 660](#) for more information.

Timeout

Parameter: Value: INTEGER or variable

Return value type: VOID

This function specifies how long, in seconds, the interaction will be held before being routed to the default destination if it has not already been routed elsewhere.

UpdateScript

Parameters: Script: STRING or variable (representing a string for the script)

Return value type: VOID

This function attaches a special key-value pair describing the selected IVR script to the interaction User Data. The key is MyScript with the database ID value of the script itself. (If the script does not exist the ID is 0.) The value for the key-value pair is the script name.

This function can be used for Voice Treatment Option (VTO) as a part of treatment objects (busy or mandatory) to provide script information required by VTO.

This function returns no errors.

UseAgentState

Parameters: String or variable (representing the Agent State)

Return value type: VOID

This function specifies the agent state URS will use instead of the default reported by Stat Server.

To use this option you must first set up URS as follows:

1. In the Annex or Options tab of the URS application, create a section called AgentStates (case sensitive).
2. Within that section, create an option for every user-defined agent state. URS can accept up to 32 options in the AgentStates section.
3. For each option, specify its value in the format:

Function[DN type]<op1>Function[DN type]<op1>...Function[DN type]<op2>number

(Format1 expression)<op3>(Format1 expression)<op3>

where

- Function[DN type] is one of the following predefined functions:
 - ready [DN type] - which returns the number of agent DNs of the specified type are in the ready state at the current moment
 - busy [DN type] - which returns the number of agent DNs of the specified type are in the busy state at the current moment

DN type for these predefined functions is the agent's DN. Types include ACDPosition, Extension, E-mail, Eaport, Cellular, Chat, Cobrowse, Fax, Voicemail, Voip, Video, and Workflow.

- op1 is an operator of either plus (+) or minus (-)
- op2 is an operator of either greater than (>), less than (<), or equal to (=)
- op3 is logical operator either or (|) or and (&)
- multiplication (*)
- division (/)
- number is zero or any positive number to evaluate the expression

For example, the agent is defined as ready if the agent has no calls on extension or position `busy[extension]+busy[acdposition] = 0`. An agent in this state will be considered not ready if the agent has at least one call on the extension or position. The agent will be considered ready in all other situations; that is if the agent has the e-mail DN busy.

Important Information

- When using the UseAgentState function, whole numbers are rounded (1.25 is counted as 1).
- Option values cannot contain spaces.
- If you want to use AgentStates for a backup URS in addition to the primary URS, create an identical AgentStates section in the backup URS.
- When using this function, you must use lowercase DN Types without “-” (hyphen). For example, use “email” not “E-mail”.
- If function CheckAgentState is set to false (see description on [page 507](#)), URS ignores any agent state, either the default one (reported by Stat Server) or the user-defined one (as described above).
- URS uses integer arithmetic in its calculations, such as for agent state and skill expression evaluation. For this reason, you must always create expressions based on integer arithmetic, not float.

UseAgentStatistics

Parameters: true or false (default): INTEGER or variable

Return value type: VOID

UseAgentStatistics is a universal function that may be used for any appropriate statistic. It can also be used for cost-based routing, which is described in the *Universal Routing 7.6 Cost-Based Routing Configuration Guide*. This function makes URS apply statistics for target selection at level of individual Agents or Places even if targets are groups of corresponding objects, such as Agent Groups or Place Groups.

VQSelected

Parameters: none

Return value type: STRING

When called as a postrouting action, this function returns the alias of the virtual queue that URS selected. If no virtual queue is specified for the target list or if the interaction has not yet been routed when the function is called, the empty string is returned.

Reporting Functions

Figure 239 shows the Function Properties dialog box with Reporting selected.

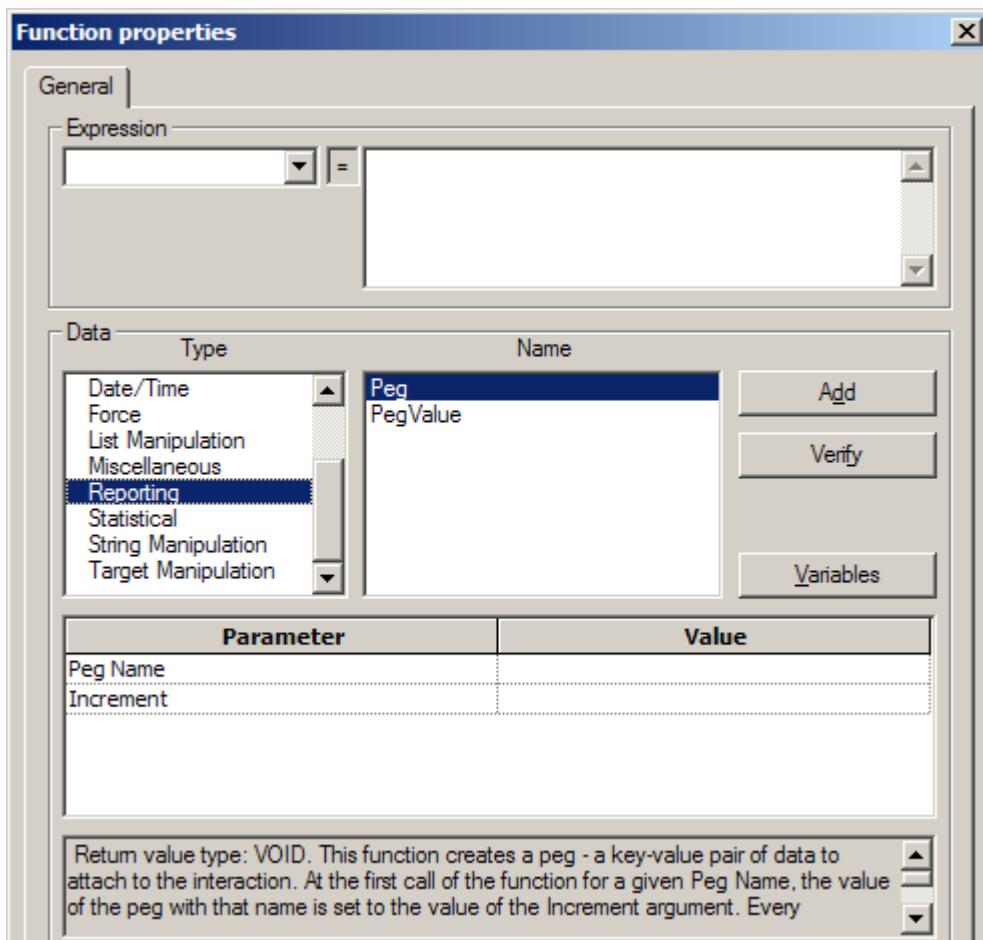


Figure 239: Function Properties Dialog Box, Reporting Functions

This section presents functions in the Reporting category (see Table 130 on page 430).

Peg

Parameters: Peg Name: STRING or variable (representing a string for the peg name)

Increment: INTEGER or variable

Return value type: VOID

This function creates a user-defined *peg*—a key-value pair of data to attach to the interaction. At the first call of the function for a given Peg Name, the value of the peg with that name is set to the value of the Increment argument. Every subsequent call with the same Peg Name adds Increment to the previous value of the peg. Before URS directs T-Server to route the interaction, it attaches to the interaction the pair with the key Peg Name and with the value of the sum of all encountered increments.

URS sends peg data along with previously attached data to a remote T-Server in the external routing dialog box (conducted by URS if the option use_extrouter is set to true).

See Chapter 6, “Automatically Attached Pegs,” [page 711](#) for more information about the peg function.

PegSF

In addition to user-defined pegs, there is a set of automatic pegs that URS can attach to an interaction in certain circumstances. All automatic pegs are always incremented by 1 with exception of PegSF, which does not increment with each interaction. Instead, a new peg is created. URS only attaches it when a Service Level routing rule is encountered in a strategy. PegSF does not provide real-time information about achieving Service Level specifications. URS calculates the peg value based on an average of the levels for the first 50 calls or every 30 seconds, whichever comes first.

Important Information

- External routing for T-Servers is now called Inter Server Call Control (ISCC)

PegValue

Parameters: Object: STRING (high-level target) or variable (representing a string for the target)

Statistic: STRING or variable (representing a string for the statistic)

Return value type: FLOAT

This function operates in the same way as SData (see [page 572](#)). In IRD, the types in the dialog box for the Object and Statistic parameters include:

- Agent

- Agent Group
- Destination Label
- Place
- Place Group
- Queue
- Queue Group
- Routing Point (virtual and real)
- Variable

For all types but Variable, you also specify the name of the object for the selected type and its location (the Stat Server name). For the Variable type, select the name of the variable only.

The following errors are possible:

- 0001 Unknown error
- 0013 Remote error
- 0014 Unknown server
- 0015 Closed server

Statistical Functions

Figure 240 shows the Function Properties dialog box with Statistical selected.

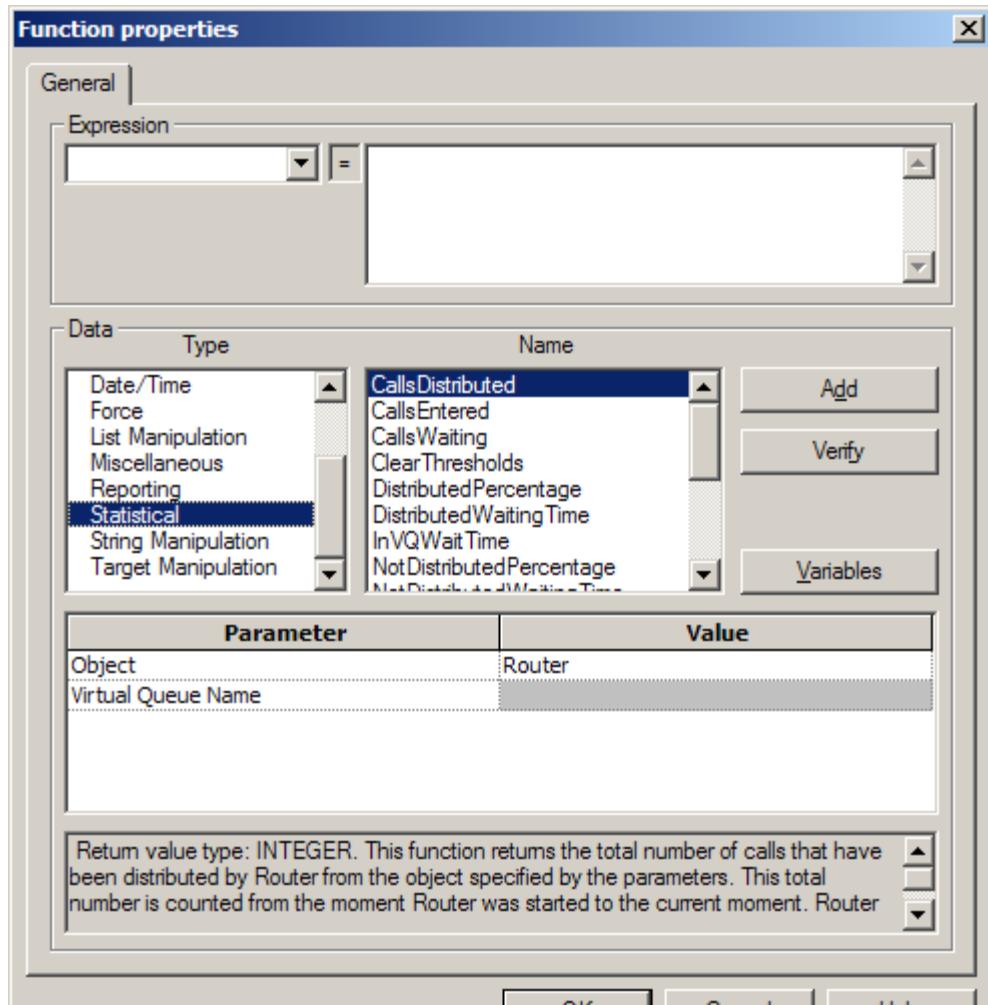


Figure 240: Function Properties Dialog Box, Statistical Functions

The **Statistical** group of functions calculate internal statistical information on the work of URS, such as how many interactions have been distributed from some point, how many have failed to be distributed, where the current interaction is placed in an internal queue, how long it takes on average for an interaction to be distributed, and so on.

Note: The Function object does not list the following predefined functions: `sdata`, `acfgdata`, `callage`, `lcfgdata`. You cannot select these functions in the Function Properties dialog box. They can only be selected in the Threshold Expression Properties dialog box. For more information on these specialized functions, see the chapter on Share Agent By Service Level Agreement Routing in the *Universal Routing 8.0 Routing Application Configuration Guide*.

CallsDistributed

Parameters: Object: Router, RoutingPoint, or VirtualQueue

Virtual Queue Name: STRING or variable (representing the string for the virtual queue) for virtual queue objects only

Return value type: INTEGER

This function returns the total number of calls that have been distributed by URS from the object specified by the parameters. This total number is counted from the moment URS was started to the current moment. URS considers an interaction distributed as soon as it sends a routing instruction to T-Server.

If the value of the Object parameter is Router, then the result is the total number of calls that URS has distributed from all routing points.

If the value of the Object parameter is RoutingPoint, then the result is the total number of calls distributed from the same routing point as the current interaction.

If the value of the Object parameter is VirtualQueue, then the result is the total number of calls distributed from the virtual queue specified by the second parameter. The value of the parameter must be a virtual queue name and can be specified in two ways:

- Enter the name manually.
- From the dialog box, click inside the Value field and select Router, RoutingPoint or VirtualQueue from the resulting Key dialog box. The values available are the configured virtual queues. Or click Variable, and then select or enter a value. The values available for the Variable type are the variables included in the Variable list.

No value for the parameter Virtual Queue Name can be specified if the Object parameter has a value of Router or RoutingPoint.

Important Information

- Since an interaction can wait in several virtual queues at the same time, it is important to understand that an interaction is considered distributed from a virtual queue if the target to which the interaction is routed is selected from that virtual queue. The same target can receive interactions from different virtual queues.

CallsEntered

Parameters: Object: Router, RoutingPoint, or VirtualQueue

Virtual Queue Name: STRING or variable (representing the string for the virtual queue) for virtual queue objects only

Return value type: INTEGER

- If the `Object` parameter has a value of `Router`, then this function returns the total number of calls for which URS has received routing requests since it started work.
- If the `Object` parameter has a value of `RoutingPoint`, then this function returns the total number of calls for which URS has received routing requests from the same routing point as that of the current interaction.
- If the `Object` parameter has a value of `VirtualQueue`, then this function returns the total number of calls that URS has placed in the virtual queue specified by the second parameter. The value of the parameter must be a virtual queue name.

If `VirtualQueue` is selected as the `Object` parameter, the value of `Virtual Queue Name` can be specified in two ways:

- Enter the name manually.
- From the dialog box, opposite `Virtual Queue Name`, click under `Value`. In the resulting Key dialog box select a Type (`Virtual Queue` or `Variable`), and then select or enter a value. The values available are for the configured virtual queues. The values available for the `Variable` type are the variables included in the Variable list.

CallsWaiting

Parameter: Target: STRING (statistical object) or variable (representing a string for a target)

Return value type: INTEGER

This function returns the number of calls currently waiting for a specified target. It takes into account all interactions waiting for the targeted skill level, both directly and indirectly (through groups). The value of the parameter must be in target format (`Name@StatServerName.Type`).

In IRD, the target types in the dialog box for this function include:

- Agent
- Agent Group
- Destination Label
- Place
- Place Group
- Queue (virtual and real)
- Queue Group
- Routing Point (virtual and real)
- Variable

For all target types but `Variable`, you also specify the name of the target for the selected type and its location (Stat Server name). For the variable target type, select the name of the variable only.

Target Parameter

When parameter Target is specified as a virtual queue, this function will return total number of calls in the virtual queue.

When parameter Target is specified as Target, this function will return the “sum of all calls” gathered from all virtual queues waiting for the same target.

Calls Waiting Target Example

A target English>5 and TechSupport=10 are commonly shared between VQ_A and VQ_B. Each virtual queue has 5 calls. Using CallsWaiting(English>5 and TechSupport=10) will return 10 calls waiting for the same target.

Note: You can combine this function with functions InVQWaitTime, PositionInQueue, and SetStatUpdate to get/post data for an IVR wait time announcement. See [page 576](#) for the details.

ClearThresholds

Parameters: none

Return value type: VOID

This function invalidates all thresholds previously set by Threshold functions. As a result, URS now considers all targets that were previously affected by Threshold functions as unconditionally ready for routing.

DistributedPercentage

Parameters: Object: RoutingPoint or VirtualQueue

Virtual Queue Name: STRING or variable (representing a string for the virtual queue) for virtual queue objects only

Return value type: INTEGER

- If the Object parameter has a value of RoutingPoint, then this function returns the percentage of distributed calls among the last 100 interactions from the routing point of the current interaction.
- If the Object parameter has a value of VirtualQueue, then the function returns the percentage of calls distributed from the specified virtual queue over the last 100 processed calls that have been placed in the virtual queue if the number of such calls is more than 100. Otherwise, the function returns the percentage of calls distributed from the virtual queue over the total number of processed calls that have been placed in the virtual queue.

When selecting a Virtual Queue object, the value of Virtual Queue Name can be specified in two ways:

- Enter the name manually.

- From the dialog box, opposite `Virtual Queue Name`, click under `Value`. In the resulting Key dialog box select a Type (`Virtual Queue` or `Variable`), and then select or enter a value. The values available are for the configured virtual queues. The values available for the `Variable` type are the variables included in the Variable list.

No value for the parameter `Virtual Queue Name` can be specified if `Object` has a value of `RoutingPoint`.

The value of this function is truncated to its integer part.

URS considers an interaction processed when either URS has sent routing instructions to T-Server or the interaction has terminated before URS has sent routing instructions, for instance, if the calling party hangs up. In the first case, the interaction is considered distributed.

Important Information

- Since an interaction can wait in several virtual queues at the same time, it is important to note that an interaction is considered distributed from a virtual queue when the target to which the interaction was routed is selected from that virtual queue. The same target can receive calls from different virtual queues.
- If no calls have been processed from the routing point or virtual queue, the function will return a value of 0.
- This function is complemented the `NotDistributedPercentage` function, described on [page 566](#).

DistributedWaitingTime

Parameters: `Object`: `RoutingPoint` or `VirtualQueue`
`Virtual Queue Name`: `STRING` or variable (representing a string for the virtual queue) for virtual queue objects only

Return value type: `INTEGER`

This function returns the average waiting time of the calls distributed from the specified object among the last 100 processed calls from the object, or among all the processed calls from the object if this number is less than 100.

If the `Object` parameter has a value of `RoutingPoint` (the routing point of the current interaction is implied), then the waiting time of a distributed interaction is interpreted as the interval between receiving a route request and sending routing instructions.

If the `Object` parameter has a value of `VirtualQueue`, the waiting time of a distributed interaction is interpreted as the interval between the moment when the interaction was placed in the virtual queue and the moment when the interaction was routed to a target selected from the virtual queue.

When selecting a virtual queue, the value of `Virtual Queue Name` can be specified in two ways:

- Enter the name manually.
- From the dialog box, opposite **Virtual Queue Name**, click under **Value**. In the resulting **Key** dialog box select a Type (**Virtual Queue** or **Variable**), and then select or enter a value. The values available are for the configured virtual queues. The values available for the **Variable** type are the variables included in the Variable list.

Important Information

- If no calls have been processed from the routing point or virtual queue, the function will return a value of 0.

GetAvgStatData

Parameters: Targets list: STRING or variable representing a comma-separated list of targets. The function “SelectTargetsByThreshold” on [page 530](#) can return a subset of this parameter.
Statistic: STRING or variable (representing a string for the statistic)

Return value type: float

In some routing scenarios, you may wish to:

- Select a targets list by comparing each target list item's statistic to a constant.
- Select a target list by comparing an input target list against the maximum, minimum, or average value of a statistic (which may, optionally, be further modified by a simple mathematical expression). The functions, GetAvgStatData, GetMaxStatData, GetMinStatData, and GetSumStatData provide the ability to do this.

GetAvgStatData calculates specified statistic for all listed targets and returns the average value of this statistic

GetMaxStatData

Parameters: Targets list: STRING or variable representing a comma-separated list of targets. The function “SelectTargetsByThreshold” on [page 530](#) can return a subset of this parameter.
Statistic: STRING or variable (representing a string for the statistic)

Return value type: float

GetMaxStatData calculates specified statistic for all listed targets and returns the maximum value of this statistic. See GetAvgStatData for additional background information.

GetMinStatData

Parameters: Targets list: STRING or variable representing a comma-separated list of targets. The function “SelectTargetsByThreshold” on [page 530](#) can return a subset of this parameter.

Statistic: STRING or variable (representing a string for the statistic)

Return value type: float

GetMinStatData calculates specified statistic for all listed targets and returns the minimum value of this statistic. See GetAvgStatData for additional background information.

InVQWaitTime

Notes: Starting with 7.6, the StatExpectedWaitingTime statistic in the formula for InVQWaitTime is replaced with `min(max(StatLoadBalance, 0), 10000)`. StatLoadBalance more accurately counts time to wait.

For virtual queues, T-Server does not propagate the queue parameter in login messages. Therefore, in order for the StatLoadBalance statistic used by the InVQWaitTime function to work with virtual queues, you must set up an association between agents and virtual queues in Configuration Manager. The process for doing this is described in the *Framework 8.1 Stat Server's User's Guide*. See the chapter on Statistical Categories, EstimWaitTime section.

Parameters: Virtual Queue

Return value type: INTEGER

Provides the expected waiting time in a virtual queue for the current call, taking position in queue into account. URS calculates an InVQWaitTime statistic using the Stat Server-provided StatLoadBalance statistic and URS’s PositionInQueue and CallsWaiting functions. Function InVQWaitTime is, in essence, a shortcut for SData:

```
Min(Max(SData[StatLoadBalance, VirtualQueue], 0), 10000) *  
PositionInqueue[VirtualQueue] / (CallsWaiting[VirtualQueue]+1).
```

Warning! Do not use InVQWaitTime in a multi-URS environment where each URS can route to the same set of targets. This is because URS #1 does not know about the activity of URS #2.

Error codes are the same as for the SData function (see [page 572](#)):

Note: You can combine this function with functions PositionInQueue, CallsWaiting, and SetStatUpdate to get/post data for an IVR wait time announcement. See [page 576](#) for the details.

Important Information

If the InVQWaitTime function is called for an interaction that is not in queue, the function returns the `Min(Max(SData[StatLoadBalance, VirtualQueue], 0), 10000)` for a given virtual queue.

NotDistributedPercentage

Parameters: `Object: RoutingPoint or VirtualQueue`
`Virtual Queue Name: STRING or variable (representing a string for the virtual queue) for virtual queue objects only`

Return value type: `INTEGER`

This function complements the DistributedPercentage function, described on [page 562](#). All the comments there apply to NotDistributedPercentage if the word *distributed* is replaced with the words *diverted but not distributed or abandoned by caller* as is done below.

Note: Diverted but not distributed interactions include those distributed from a particular object (`RoutingPoint` or `VirtualQueue`) by means other than being routed by URS or abandoned by the caller.

- If the `Object` parameter has a value of `RoutingPoint`, then this function returns the percentage of diverted but not distributed or abandoned by caller calls among the last 100 interactions from the routing point of the current interaction.
- If the `Object` parameter has a value of `VirtualQueue`, then the function returns the percentage of diverted but not distributed or abandoned by caller calls from the specified virtual queue over the last 100 processed calls that have been placed in the virtual queue if the number of such calls is more than 100. Otherwise, if the number of such call is less than 100, the function returns the percentage of diverted but not distributed or abandoned by caller calls from the virtual queue over the total number of processed calls that have been placed in the virtual queue.

Important Information

- Since an interaction can wait in several virtual queues at the same time, it is important to note that an interaction is considered diverted but not distributed from a virtual queue when the target to which the interaction

was routed is selected from that virtual queue, but the interaction is diverted but not distributed. The same target can receive calls from different virtual queues.

- If no calls have been processed from the routing point or virtual queue, the function will return a value of 0.

When the `Object` parameter is equal to `Virtual Queue Name`, an interaction is considered diverted but not distributed from a virtual queue if it is routed to a target selected from another virtual queue, even though the target might be common to both virtual queues.

In particular, if at least one interaction has been processed from the indicated object, then the sum of `DistributedPercentage` and `NotDistributedPercentage`, if calculated at the same time for the same object, will equal 100 or 99 (99 is possible because of a truncation error). If no calls have been processed from the indicated `Object`, then the sum of `DistributedPercentage` and `NotDistributedPercentage`, if calculated at the same time for the same object, will equal 0 because both functions are programmed to return 0 in this case.

When selecting a virtual queue, the value of `Virtual Queue Name` can be specified in two ways:

- Enter the name manually.
- From the dialog box, opposite `Virtual Queue Name`, click under `Value`. In the resulting Key dialog box select a Type (`Virtual Queue` or `Variable`), and then select or enter a value. The values available are for the configured virtual queues. The values available for the `Variable` type are the variables included in the Variable list.

NotDistributedWaitingTime

Parameters: `Object: RoutingPoint or VirtualQueue`
`Virtual Queue Name: STRING or variable (representing a string for the virtual queue) for virtual queue objects only`

Return value type: `INTEGER`

This function returns the average waiting time of nondistributed interactions from the specified object among the last 100 processed calls from the object, or among all the processed calls from the object if this number is less than 100.

If the `Object` parameter has a value of `RoutingPoint` (the routing point of the current interaction is implied), then the waiting time of a nondistributed interaction is interpreted as the interval between receiving a route request and receiving an event indicating that the interaction has been terminated.

If the `Object` parameter has a value of `VirtualQueue`, the waiting time of a nondistributed interaction is interpreted as the interval between the moment when the interaction is placed in the virtual queue and the moment when the interaction is diverted from the virtual queue by URS.

When selecting a virtual queue, the value of the `Virtual Queue Name` can be specified in two ways:

- Enter the name manually.
- From the dialog box, opposite `Virtual Queue Name`, click under `Value`. In the resulting `Key` dialog box select a Type (`Virtual Queue` or `Variable`), and then select or enter a value. The values available are for the configured virtual queues. The values available for the `Variable` type are the variables included in the Variable list.

Important Information

- If no interactions are processed from the routing point or virtual queue, the function will return a value of 0.
- There is no a priori connection between the values returned by `DistributedWaitingTime` and `NotDistributedWaitingTime`.

PositionInQueue

Parameter: `Target`: STRING (statistical object) or variable (representing a string for the target)

Return value type: INTEGER

This function returns the position of the current interaction among the calls waiting for the target specified as an argument. It takes into account all interactions waiting for the targeted skill level, both directly and indirectly (through groups). The interaction with the lowest position (1) is distributed to the target first if the target becomes available.

The function returns a value of 0 if and only if the current interaction is not waiting for the specified target.

In IRD, the target types in the dialog box for this function include:

- Agent
- Agent Group
- Destination Label
- Place
- Place Group
- Queue (virtual and real)
- Routing Point (virtual and real)
- Variable

For all target types but `Variable`, you also specify the name of the target for the selected type and its location (Stat Server name). For the `Variable` target type, select the name of the variable only.

The string value of the parameter must be in target format (`Alias@StatServerName.Type`).

Target Parameter

- When parameter `Target` is specified as a virtual queue, this function will return the position of current call among others in the virtual queue.
- When parameter `Target` is specified as `Target`, this function will return the “true position” of current call among all others calls in various virtual queues waiting for the same target.

PositionInQueue Target Parameter Example

A target `English>5` and `TechSupport =10` are commonly shared between `VQ_A` and `VQ_B`. Using `PostionInQueue(English>5 and TechSupport=10)` will return the “true position” of the current call among all other calls waiting in both `VQ_A` and `VQ_B`. The “true position” of the current call among other calls is calculated by ordering calls from highest to lowest priority and, if equal priority, rank by age of interaction among calls waiting on both `VQ_A` and `VQ_B`.

Note: You can combine this function with functions `InVQWaitTime`, `CallsWaiting`, and `SetStatUpdate` to get/post data for an IVR wait time announcement. See [page 576](#) for the details.

ResetStatAdjustment

Parameters: `Target`: STRING (statistical object) or variable (representing a string for the target)

`Statistic`: STRING or variable (representing a string for the statistic)

Return value type: VOID

This function cancels any adjustment that may have been set for a statistic for a given target (see `SetStatAdjustment` on [page 574](#) for a discussion of statistic adjustments).

When selecting `Target`, the target types in the dialog box include:

- Agent
- Agent Group
- Destination Label
- Place
- Place Group
- Queue
- Queue Group
- Routing Point (virtual and real)
- Variable

For all types but `Variable`, you also specify the name of the object or statistic for the selected type and its location (Stat Server name). For the `Variable` type, select the name of the variable only.

The string value of the `Target` parameter must be in target format (`Alias@StatServerName.Type`). A string constant in target format does not need to be enclosed in quotation marks.

The parameter, `Statistic` must evaluate the name of a statistic defined in the strategy or of a predefined statistic.

When selecting a `Statistic` parameter, the value of the statistic can be specified in two ways:

- Enter the statistic value manually.
- From the dialog box, opposite `Statistic`, click under `Value`. In the resulting `Key` dialog box select `Statistic` or select `Variable`. Then select or enter a value. The values available for `Statistic` are for the configured Statistics. The values available for the `Variable` type are the variables included in the `Variable` list.

RvqData

Parameters: `Label`: STRING (label of interactions within specific internal routing queue)

Data: INTEGER (type of requested metric)

Return value type: FLOAT

This function returns the metric information, such as expected waiting time, position in queue, etc., about interactions within a specific internal routing queue, which represents the internal URS structure for interactions waiting for the same targets. In order to retrieve metrics:

- Virtual Queue must be associated with a target. The Routing target can be specified in Routing Objects, SelectDN function, `<queue.submit>` action element (for scxml applications).
- Interactions must be labeled. There are two ways to label interactions placed into an internal routing queue:
 - Using the `SetQueueLabel` function, [page 548](#). This method is available within the IRD strategy only.
 - Providing the label as a prefix of Virtual Queue, in the form of `[label]VirtualQueue`. This method is available within the IRD strategy and SCXML Application and takes precedence over the first method.

The following metrics are supported:

- `RVQ_DATA_AHT` (numeric value is 1; short name is `aht`) - Average handling time, in seconds, for the processing of a single interaction.
- `RVQ_DATA_POSITION` (numeric value is 2; short name is `pos`) - Position of interaction in internal routing queue.

- RVQ_DATA_QUEUE_LEN (numeric value is 3; short name is qlen) - Number of interactions inside the internal routing queue.
- RVQ_DATA_EWT (numeric value is 4; short name is ewt) - Expected waiting time, in seconds, for the interaction to being answered by agent. This metric is calculated as multiplication of RVQ_DATA_AHT and RVQ_DATA_POSITION.
- RVQ_DATA_QUEUE_EWT (numeric value is 5; short name is qewt) - Expected waiting time, in seconds, for all interactions in the internal routing queue to be answered by agent. This metric is calculated as multiplication of RVQ_DATA_AHT and RVQ_DATA_QUEUE_LEN+1.
- RVQ_DATA_QUIT_RATE (numeric value is 6; short name is quit) - Time interval, in seconds, between the subsequent distribution of interactions from the internal routing queue.
- RVQ_DATA_ALL_AGENTS (numeric value is 7; short name is size) - Number of all agents associated with internal routing queue.
- RVQ_DATA_REG_AGENTS (numeric value is 8; short name is insize) - Number of logged in agents associated with internal routing queue.
- RVQ_DATA_WT (numeric value is 9; short name is wt) – Time, in seconds, the interaction is waiting in this internal routing queue.
- RVQ_DATA_MIN_EWT (numeric value is 10; short name is minewt) - Minimal expecting waiting time, in seconds. This metric ignores the provided label and returns the minimal value of RVQ_DATA_EWT within all internal routing queues the interaction is associated with.
- RVQ_DATA_AGR_EWT (numeric value is 11; short name is agrewt) - Aggregative expecting waiting time, in seconds. This metric ignores the provided label and returns a combined value of RVQ_DATA_EWT within all internal routing queues the interaction is associated with. Combined value is calculated using following formula:

$$\text{AGR_EWT} = \frac{1}{\sum_{\text{all internal queues}} \frac{1}{\text{EWT}}} |$$

- RVQ_DATA_PRIORITY (numeric value is 12; short name is priority) – Interaction priority in this internal routing queue.
- RVQ_DATA_TIME (numeric value is 13; short name is time) – Timestamp, in seconds, the interaction was placed into internal routing queue.
- RVQ_DATA_ID (numeric value is 14; short name is id) - Identification of interaction placement into this internal routing queue.

The short metric names are used in:

- generic expressions as part of treatment parameters in the form {rvqdata[label, aht]} is substituted in the T-Server request with a corresponding value of the average handling time.

- REST web API method `rvqdata`. For details, see: Appendix C: URS-Behind Solution, in the section “Supported Methods” on [page 815](#).

To obtain some metrics related to average handling time (AHT), URS processes data for all agents associated with internal routing queues to calculate the AHT of a single interaction for each agent and for each media. Calculations are based on the average of the last 10 completed interactions of a given media. While URS has not collected this information, the hardcoded values of 150 seconds for voice, and 300 seconds for all other medias are used.

For voice, AHT values can be overwritten with the options `agent_att` ([page 629](#)) and `use_agent_att` ([page 674](#)).

URS first checks if `agent_att` and `use_agent_att` options are defined on Virtual Queue object, then on URS object. If options not configured, then hardcoded values will be used.

The following error messages are possible:

- `0018 Unknown object` – when internal routing queue with label is not found
- `0017 Invalid request` – when unknown metric is requested

SData

Note: Do not confuse the SData function discussed below with the (lowercase) `sdata` function used for defining threshold expressions when implementing Share Agent by Service Level Agreement routing. You can only specify the `sdata` function in IRD’s **Threshold Expression Properties** dialog box, whereas you specify the SData function in the **Function Properties** dialog box. If you need information on using the specialized `sdata` function, consult the chapter on Share Agent by Service Level Agreement Routing in the *Universal Routing 8.0 Routing Application Configuration Guide*.

Parameters: `Target`: STRING (statistical object) or variable (representing a string for the target Agent, Agent Group, Campaign, Campaign Group, Destination Label, Place, Place Group, Queue, Queue Group, Interaction Queue, Routing Point)
`Statistic`: STRING or variable (representing a string for the statistic)

Return value type: FLOAT

This function returns the value of a statistic for a specified target. It is used in expressions with Attach and Assign. SData can be used to tell URS to return the number of interactions waiting, so that if the target is not available, the caller will hear the IVR announce the number of interactions ahead of him. The values of the `Target` and `Statistic` parameters are the same as for the parameter values of the function `ResetStatAdjustment` on [page 569](#).

Like the ExpandGroup function, SData can suspend the strategy. This is done if the required statistic is not open in Stat Server. URS requests Stat Server to open the statistic and then the strategy resumes.

When defining the Target parameter, the target types in the dialog box include:

- Agent
- Agent Group
- Campaign
- Destination Label
- Interaction Queue
- Place
- Place Group
- Queue (virtual and real)
- Queue Group
- Routing Point (virtual and real)
- Variable
- Campaign Group

SData can also be applied to targets of the DN type. Since IRD has no predefined list of targets of this type, it does not provide this type of target in the dialog box. You must manually enter a DN-type target.

For all types but Variable, you also specify the name of the object for the selected type and its location (the Stat Server name). For the Variable type, select the name of the variable only.

Note: For customers who want to specify how SData behaves if the connection is lost between URS and Stat Server or Stat Server and T-Server such that an invalid statistic message is returned, release 6.5 includes the `function_compatibility` option ([page 642](#)). When the option value is left as 6.1 (default) and the connection between Stat Server and URS is broken, SData returns `0` and the interaction flows to the green port. When a value of 6.5 is specified and the connection between Stat Server and URS is broken, SData also returns `0` but the interaction flows to the red port of the function object rather than the green port. Otherwise, SData functions as it did in previous releases when the connection between Stat Server and URS is broken; interactions flow to the green port.

When selecting the Statistic parameter, the value of the statistic can be specified in two ways:

- Enter the statistic value manually.
- From the dialog box, opposite Statistic, click under Value. In the resulting Key dialog box, select a Type, Statistic or Variable, and then select or enter a value. The values available are for the configured Statistics. The values available for the Variable type are the variables included in the Variable list.

If the specified statistic is neither on the list of predefined statistics nor defined in the strategy, the function returns a value of `0`.

The following error messages are possible:

- `0001 Unknown error`—for example, invalid target specification
- `0013 Remote error`—Stat Server returns an error
- `0014 Unknown server`—specified Stat Server cannot be found
- `0015 Closed server`—specified Stat Server is closed

Use Case

During month end when billing questions to a call center peak, the objective is to get an agent to respond to a call ASAP, but to get a skilled agent whenever possible. The routing plan is:

1. Billing interactions are routed to a target list with 1 agent group:
`AgentGroup2.VQ2` is the virtual queue that collects queue related statistics.
This group of agent is specially trained.
2. General interactions are routed to a target list with two agent groups:
`AgentGroup1` and `AgentGroup2`. `VQ1` is the virtual queue that collects queue related statistic for both target groups.
3. If the predicted wait time of `AgentGroup2` (in other words, billing) is over a threshold, strategy developer queues the interactions directly to `VQ1` instead of waiting for `AgentGroup2` to exceed a threshold before queueing to `AgentGroup1`.

Note: Function `InVQWaitTime` (see [page 565](#)) is a shortcut for `SData`.

SDataInTenant

The IRD `SDataInTenant` function is similar to the `SData` function (see [page 572](#), but different in that it contains an additional parameter which represents the tenant name. The tenant name is the first parameter and statistics are requested in the context of this tenant. If the tenant name is not provided the `SDataInTenant` behaves exactly like the `SData` function.

IRD does not include this function in the check-integrity schema (whether or not the tenant name is provided).

SetStatAdjustment

Parameters:

- `Target`: STRING (statistical object) or variable (representing a string for the target)
- `Statistic`: STRING or variable (representing a string for the statistic)
- `Sign`: `+, -, /, *`

Value: FLOAT

Return value type: VOID

This function enforces an adjustment of the values of a specified statistic for a particular target. The statistic is adjusted as follows:

- If **Sign** equals +, then **Value** is added to the result reported by Stat Server.
- If **Sign** equals -, then **Value** is subtracted from the result reported by Stat Server.
- If **Sign** equals *, then **Value** is multiplied by the result reported by Stat Server.
- If **Sign** equals /, then the result reported by Stat Server is divided by **Value**.

The values of the **Target** and **Statistic** parameters are the same as the parameter values of the **ResetStatAdjustment** function on [page 569](#). The adjustment does not apply to the result of explicit Stat Server queries by the functions **SData** and **PegValue**. It is only used for thresholds, statistical interaction distribution, or when a statistic is supplied as an argument to the function **SelectDN**.

Once set, an adjustment can be annulled by a call to the function **ResetStatAdjustment**, described on [page 569](#).

SetStatUpdate

Parameters: **Key:** STRING

Statistic: STRING or variable (representing a string for the statistic)

Target: STRING (statistical object) or variable (representing a string for the target)

Return value type: VOID

The **SetStatUpdate** function allows you to attach and periodically update statistical data to an interaction while the interaction waits for an available target without leaving the target selection object. Information is attached as interaction User Data with the provided **Key**. Any statistic that can be used with **SData** function is acceptable plus three additional statistics: **PositionInQueue**, **CallsWaiting**, **InVQWaitTime**. Update period is 30 seconds.

- If the **Key** parameter is empty, URS stops periodic statistical updates of the current interaction for all previously specified statistics.
- If the **Statistic** or **Target** parameters are empty, URS stops periodically updating the current interaction Statistic associated with the provided **Key**.

Target Types

In IRD, the Target types in the dialog box for SetStatUpdate includes:

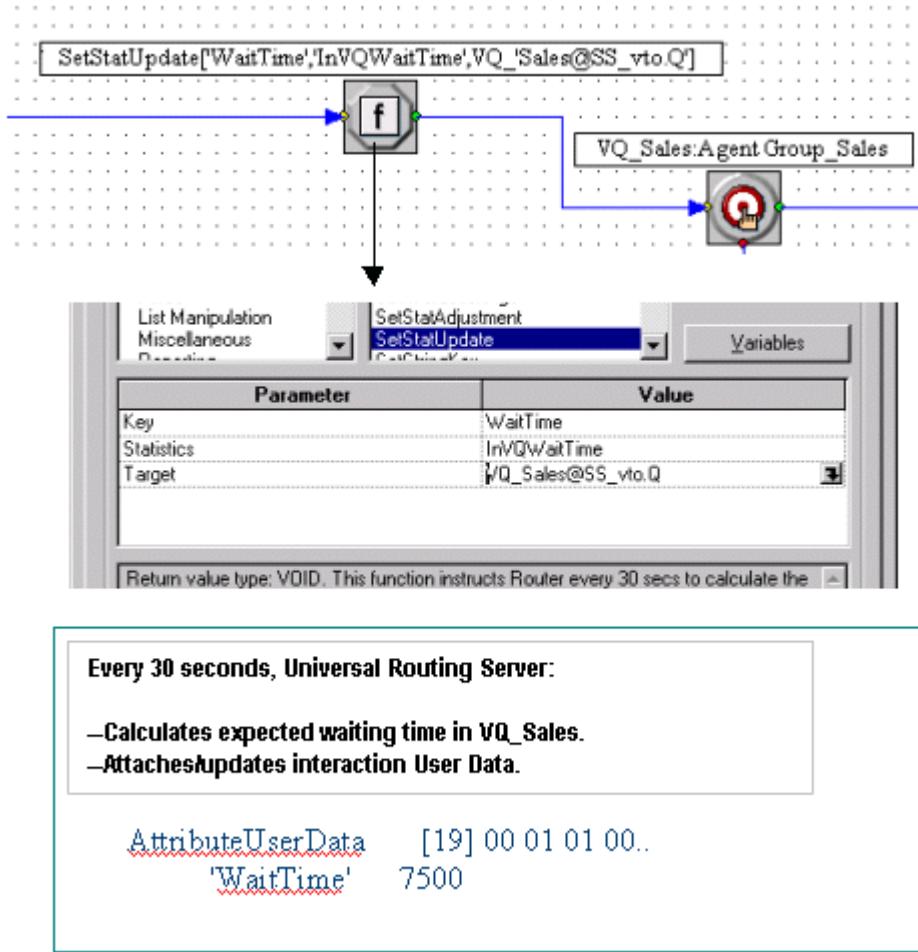
Agent

Agent Group (virtual or real)
Destination Label
Place
Place Group
Queue (virtual or real)
Queue Group
Routing Point (virtual and real)
Variable, representing any of listed above targets

Use Case

You can combine functions InVQWaitTime, PositionInQueue, CallsWaiting, and SetStatUpdate to get/post data for an IVR wait time announcement as follows:

1. First use function InVQWaitTime (Virtual Queue) (see [page 565](#)) to obtain expected the waiting time in the virtual queue for the current call, taking into account call position in the virtual queue.
2. Next, use function PositionInQueue (Target) (see [page 568](#)) to obtain the position of the call waiting for the target. Parameter Target can be a virtual queue or another target (see “PositionInQueue Target Parameter Example” on [page 569](#) for details).
3. Use function CallsWaiting (Target) (see [page 561](#)) to obtain the total number of calls waiting within for the target. Parameter Target can be a virtual queue or another target (see “Calls Waiting Target Example” on [page 562](#) for details).
4. Then use function SetStatUpdate(Key, Statistic, Target) to attach and periodically update the statistical data (see [Figure 241](#)).

**Figure 241: SetStatUpdate Function in Strategy**

- Frequency of update is every 30 seconds while waiting for the target.
- Key is a string indicating what statistic is attached.
- Statistic is any stat accepted by SData function (see [page 572](#)) and functions:
 PositionInQueue
 CallsWaiting
 InVQWaitTime

SetThresholdEx

Note: Threshold conditions imposed on targets with the SetThresholdEx function are supported only by the Selection routing object.

Parameters: Target: STRING (statistical object) or variable (representing a string for the target)

Statistics: STRING or variable

Threshold Value: INTEGER or variable

Condition: ReadyIfLess, ReadyIfGreater (default),
ReadyIfNotLess, ReadyIfNotGreater

Return value type: VOID

This function is used to configure queue or routing point statistical thresholds. The specified queue or routing point is assumed by the strategy to be unavailable to receive interactions if the specified statistic is greater than the specified value (if the selected Condition is ReadyIfLess) or if the specified statistic is less than the value (if the selected Condition is ReadyIfGreater). Every new threshold invalidates any previously set threshold.

The threshold may be placed anywhere in a strategy, but if more than one threshold is specified for the same queue, the last one is assumed to be the valid threshold.

When defining the Target parameter, the target types in the dialog box include:

- Agent
- Agent Group
- Destination Label
- Place
- Place Group
- Queue
- Queue Group
- Routing Point (virtual and real)
- Variable

For all types, you also specify the name of the object or statistic for the selected type and its location (Stat Server name).

When selecting a Statistics parameter, the value of the statistic can be specified in two ways:

- Enter the statistic value manually.
- From the dialog box, opposite Statistics, click under Value. In the resulting Key dialog box, select a Type, Statistic or Variable, and then select or enter a value. The values available are for the configured Statistics. The values available for the Variable type are the variables included in the Variable list.

Important Information

- When working with SetThresholdEx, the conditions ReadyIfLess and ReadyIfGreater actually operate on a basis of “less than” and “greater than” respectively.
- Although designed to be applied to queue or routing point types of targets, you can also apply it Agent, Place, Agent Group, or Place Group target types as well.

Note: When you select the Agent, Place, Agent Group, and/or Place Group target types for routing, and you set thresholds using the SetThresholdEx function, routing decisions are made according to the threshold value and target status sent by Stat Server.

- When the Threshold limit is reached, no more calls are routed to the target, even if Stat Server reports that the target is in the Ready state.

Note: You can specify thresholds for targets directly for the targets themselves (not separately, as with the SetThresholdEx function). To do so, add a prefix in the format `Threshold{Statistic op value}` in front of the target definition. This applies the specified threshold to the target.

UseCapacity

Parameter: Condition: `OnStatError`, `Never`, `Only`, or variable (representing a string for the condition)

Return value type: `VOID`

This function instructs URS on the condition for using configured Capacity Tables for computing statistical values. If `OnStatError` is supplied to the parameter, URS will compute a statistical value from Statistical Tables whenever the attempt to obtain the corresponding value from Stat Server results in an error. The other two options are either always use Statistical Tables for such values or never use them.

Statistics That Can Be Modeled

- `StatAgentsAvailable`—number of agents in ready state in a group (Agent Group statistic).
- `StatAgentsTotal`—number of agents logged on in a group (Agent Group statistic). When Capacity Tables are used, this statistic is not calculated but retrieved from the agent group Capacity Table.
- `StatAgentsBusy`—number of busy agents in a group (Agent Group statistic).
- `StatCallsInQueue`—number of calls in a queue (Queue statistic).
- `StatCallsAnswered`—number of calls answered (Agent Group statistic).
- `StatCallsCompleted`—number of calls completed (Agent Group statistic).
- `StatAverageHandlingTime`—average handling time of an interaction (Agent Group statistic). When Capacity Tables are used, this statistic is not calculated but retrieved from the agent group Capacity Table.

- StatLoadBalance—load-balancing statistic (Queue statistic). This was previously handled by StatEstimatedWaitingTime in Interaction Router 5.1x and Enterprise Routing and Network Routing 6.0.

Since URS recognizes modeled statistics by name, the above names are obligatory.

Configuring Capacity Tables

A Capacity Table is configured in Configuration Manager as a Statistical Table object of Capacity Table type. See the Configuration Manager documentation for more details on the process of configuration. The Capacity Table is associated with an Agent Group and must be specified inside the Advanced properties of the group. The same Capacity Table can be associated with more than one Agent Group.

Statistical Days

The Capacity Table must contain Statistical days. The only relevant values in a Statistical Day for the purpose of modeling statistics are:

- Date—Day of Year (highest precedence), or Day of Week, or Any (lowest precedence). On a particular date, modeling information will be taken from the Statistical Day for that date if present in the Configuration Database, otherwise from a Statistical Day corresponding to the day of the week; the default Statistical Day Any will be used if neither the date nor the corresponding day of the week has a Statistical Day configured.
- Business Day Start—specifies the time of day when the Statistical Day data can first be used. The count of the Time Intervals begins at the time specified as the Start of the Business Day.
- Business Day End—specifies the time of day when the Statistical Day data will no longer be used.
- Interval Length—specifies the length of the intervals into which the day is partitioned. The first interval starts at the time specified in the Start of the Business Day. The last interval can be shorter than the specified length.
- Statistical Value 1 and Statistical Value 2 for each Interval of the day—Value 1 is interpreted as the number of agents logged on in the group, Value 2 is interpreted as the average handling time.

All the other properties of Statistical Days are irrelevant for the purpose of calculating modeled statistics.

Note: The same Statistical Day can belong to more than one Statistical Table, either of type Capacity Table or of type Quota Table.

String Manipulation Functions

Figure 242 shows the Function Properties dialog box with String Manipulation selected.

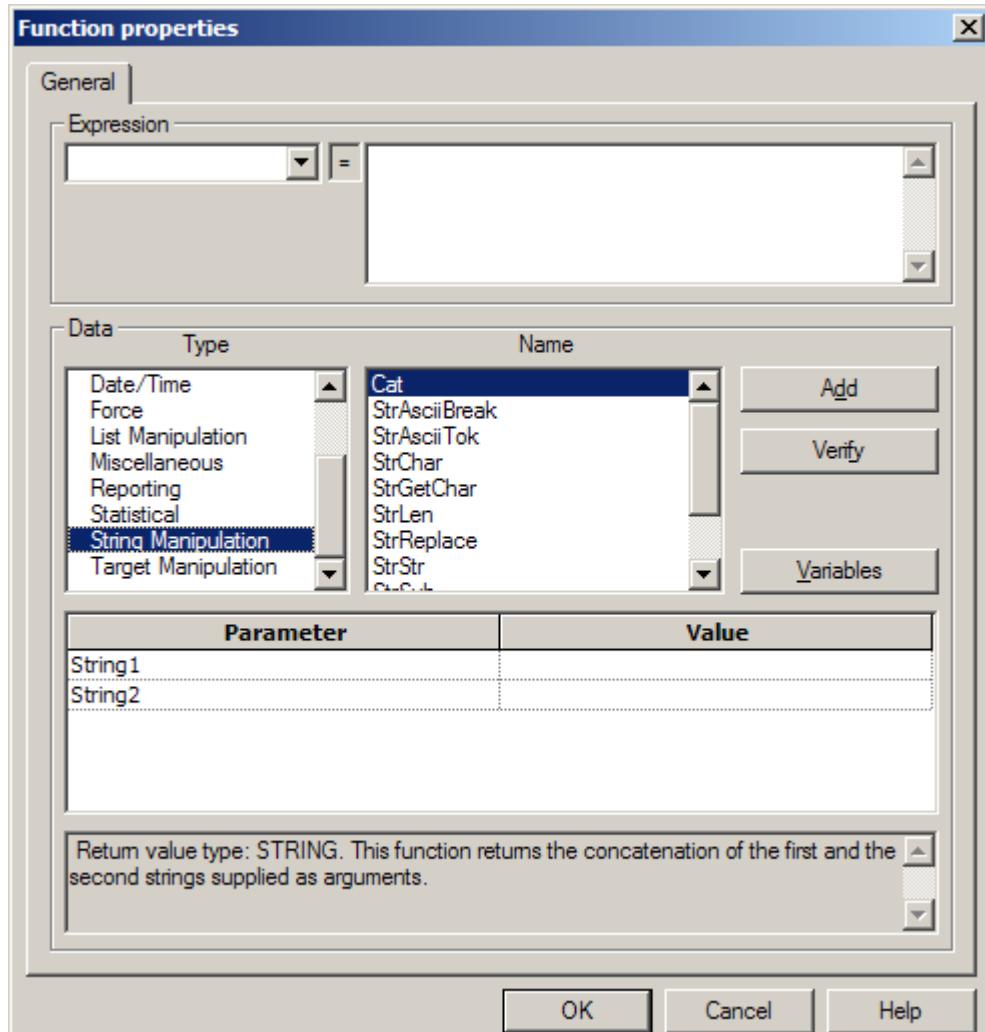


Figure 242: Function Properties Dialog Box, String Manipulation Functions

These functions make it easy to find characters and substrings within strings and to parse out substrings enclosed by prescribed separators.

Bytes

Parameters: String: STRING, Radix: 10 or 16

Return value type: STRING

This function returns the string, presenting space-separated numerical values of every character from the input string.

If the `Radix` parameter has the value of 10, the returned value is represented in the decimal numeral system.

If the `Radix` parameter has the value of 16, the returned value is represented in the hexadecimal numeral system.

Cat

Parameters: `String1`: STRING or variable (representing a string)
`String2`: STRING or variable (representing the string)

Return value type: STRING

This function returns the concatenation of the first and second strings supplied as arguments.

StrAsciiBreak

Parameters: `String`: STRING or variable (representing the string)
`Char Set`: STRING or variable (representing a string for the character set)
`From Index`: INTEGER or variable

Return value type: INTEGER

This function returns the index of the first occurrence in `String` of any of the characters in `Char Set`, such that its index is greater than or equal to the value of the `From Index` parameter. The index of the first character in a string is 1. If the `From Index` parameter is negative, the returned value is the same as if `From Index` parameter had been 1; that is, the search proceeds from the first character of the string.

If none of the characters in `Char Set` occurs in `String`, if `Char Set` is the empty string, or if the `From Index` parameter has a value greater than the length of `String`, the function returns 0. Since 0 is not a valid index, it will only be returned in these three cases. For example:

- `StrAsciiBreak['asdfg', 'sf', 2]` returns 2
- `StrAsciiBreak['asdfg', 'sf', 3]` returns 4
- `StrAsciiBreak['asdfg', 'h', 2]` returns 0
- `StrAsciiBreak['asdfg', '', 2]` returns 0

StrAsciiTok

Parameters: `String`: STRING or variable (representing a string)
`Char Set`: STRING or variable (representing a string for the Char Set)
`From Index`: INTEGER or variable

Return value type: STRING

The purpose of this function is to parse out a string value from a string consisting of multiple substrings delimited by characters functioning as separators; the separators are the characters occurring in `Char Set`.

If the `From Index` parameter is a positive integer, the function returns the substring of `String` beginning with the character given by the `From Index` parameter if this character is not a separator, or at the first subsequent non-separator character if the character given by the `From Index` parameter is a separator, and ending with the character immediately preceding the next occurrence of a separator or with the last character of `String` if no separator occurs after the initial character of the substring.

The index of the first character in a string is 1.

If the `From Index` parameter has a value greater than the length of `String` or if all the characters in `String` starting with the `From Index` parameter are separators, the empty string is returned. The empty string will only be returned in these two cases. For example:

- `StrAsciiTok['toy, pan, pot', ',', 1]` returns 'toy'
- `StrAsciiTok['toy, pan, pot', ',', 2]` returns 'oy'
- `StrAsciiTok['toy,,, pan', ',', 4]` returns 'pan'
- `StrAsciiTok['rty', ':', 5]` returns ''
- `StrAsciiTok['dfgyyyu', 'yu', 4]` returns ''

Every time the function `StrAsciiTok` is called, it stores in memory the index in `String` of the last character of the obtained substring.

If the `From Index` parameter is 0 or a negative number, at the first call of the function, the returned value is the same as if the `From Index` parameter had been 1; that is, the search proceeds from the first character of the string. If the `From Index` parameter is 0 or a negative number and the function has been called previously, then the function replaces the parameter `From Index` by one plus the previously stored index of the last character of the last substring returned. This makes it possible to easily parse out consecutive elements of a list, while ignoring the intervening separators.

For instance, calling `StrAsciiTok['toy,,, pan', ',', 0]` three consecutive times will return `toy` the first time, `pan` the second time, and '' (the empty string) the third time.

Important Information

- Use the zero value for the `From Index` parameter only when the immediately preceding call of `StrAsciiTok` uses the same values for `String` and `Char`.

StrChar

Parameters: `String`: STRING or variable (representing the string)

Character: STRING or variable (representing a string for the character)

From Index: INTEGER or variable

Return value type: INTEGER

This function returns the index of the first character of the first occurrence of **Character** as a substring of the tail segment of **String** starting at **From Index**.

The index of the first character in a string is 1. If the **From Index** parameter is negative, the returned value is the same as if the **From Index** parameter had been 1; that is, the search proceeds from the first character of the string.

The function returns 0 if and only if the tail segment of **String** starting at **From Index** does not contain **Character** as a substring, or the **From Index** parameter is greater than the length of **String** and **Character** is not empty.

Important Information

- The parameter name *Character* may prove misleading. This parameter can contain an arbitrary string to search for.
- If **Character** is the empty string, the function returns the value of the **From Index** parameter even if the **From Index** parameter is greater than the length of **String**. For example:
 - `StrChar['ababa', 'ab', 1]` returns 1
 - `StrChar['ababa', 'ab', 2]` returns 3
 - `StrChar['ababa', 'c', 2]` returns 0
 - `StrChar['abcdef', 'c', 4]` returns 0
 - `StrChar['abcdef', 'c', 8]` returns 0
 - `StrChar['abcdef', '', 8]` returns 8

StrGetChar

Parameters: **String:** STRING or variable (representing the string)

Index: INTEGER or variable

Return value type: STRING

This function returns the character with a specified index in a string. If **Index** is negative or greater than the length of **String**, the function returns the empty string.

StrLen

Parameter: **String:** STRING or variable (representing the string)

Return value type: INTEGER

This function returns the length of the string supplied as an argument; that is, the number of characters in the string.

StrNextTokInd

Parameter: None

Return value type: INTEGER

This function is complimentary to the StrAsciiTok function. It returns the position in the search string as set by the last invocation of the StrAsciiTok function. This value can later be used as an explicit value for the From Index parameter of StrAsciiTok.

This function allows retrieval and storage of an index after the last invocation of StrAsciiTok. Next time, StrAsciiTok can be called with this index regardless of whether or not StrAsciiTok was called in between. The pair of functions StrAsciiTok and StrNextTokInd allows going through several strings simultaneously.

StrReplace

Parameters: Source: STRING

To Find: STRING or variable (representing a searched substring)

Replace With: STRING or variable (representing a substring to replace)

Return value type: STRING

This function returns a string obtained from a Source string by replacing every To Find fragment with Replace With one. For example, to convert string 04/25/03 into 042503 (remove slashes from date string) StrReplace['04/25/03', '/', ''] can be used.

StrStr

Parameters: Source: STRING or variable (representing a string for Source)

To Find: STRING or variable (representing a string for To Find)

Return value type: STRING

This function finds the first occurrence of the string To Find as a substring of Source and returns the tail segment of Source starting with that occurrence. If To Find is not a substring of Source, the empty string is returned.

StrSub

Parameters: Source: STRING or variable (representing a string for the Source)

First: INTEGER or variable

Last: INTEGER or variable

From The END: False, True, or variable

Return value type: STRING

This function returns the substring of `Source` beginning with the character with index `First` and ending with the character with index `Last`. If the `From The END` parameter has a value of `true`, then the indices `First` and `Last` are counted from the end of the string `Source`.

The index of the first character in a string is 1.

StrTargets

Parameters: `String: STRING or variable (representing the string for the target)`

Return value type: `STRING`

This function facilitates creating a comma-separated list of targets for use as input parameters for the Genesys-provided utility subroutines described in the Samples chapter in the *Universal Routing 8.1 Deployment Guide*.

Prior to 7.6, you could do this with the `Cat[]` function, but this function required you to manually enter the names of all targets. Function `StrTargets` make this task easier. When you use this function in a strategy, for every function parameter, IRD provides a `TARGET` dialog box. [Figure 243](#) shows the `TARGET` dialog box after selecting the `Agent Group` target type.

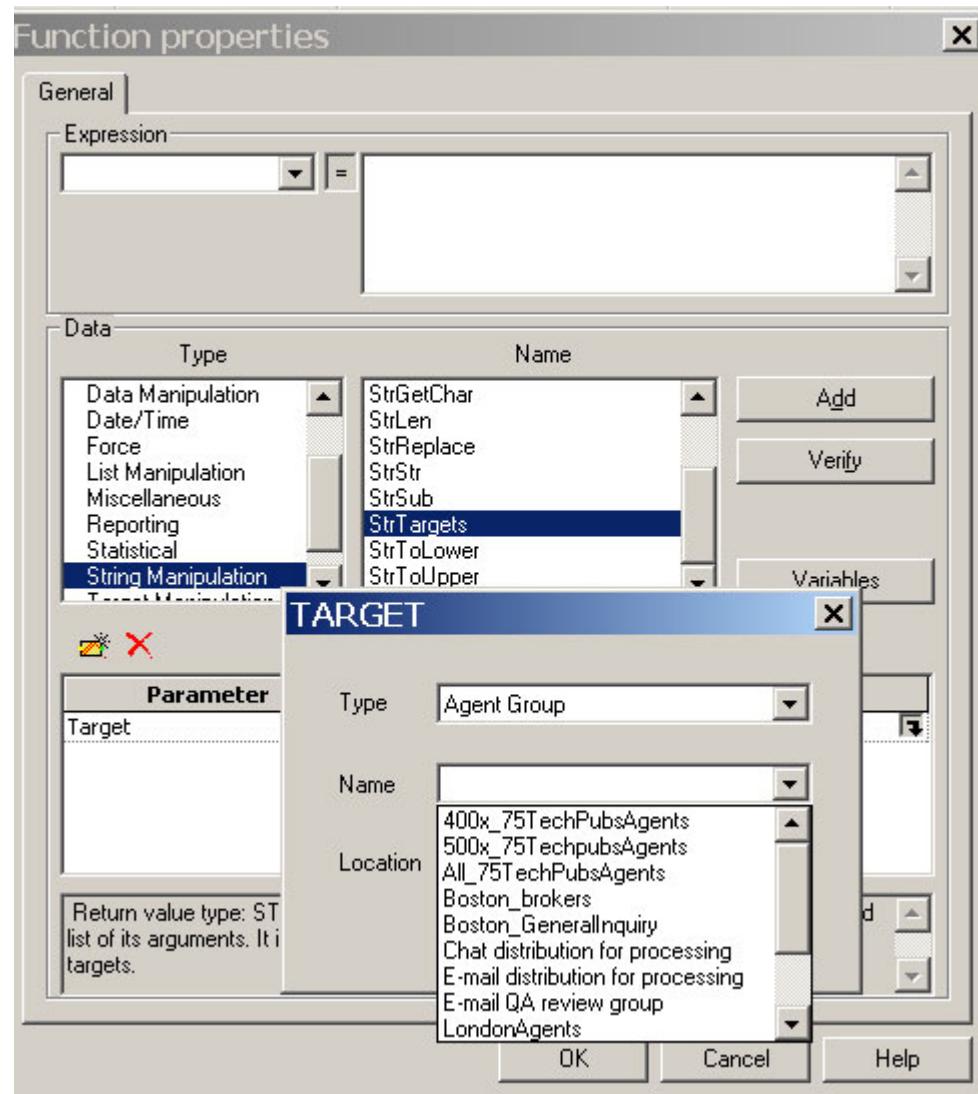


Figure 243: Function StrTargets and TARGET Dialog Box

This frees you from the need to manually type in target names. For example, as shown in [Figure 243](#), after selecting the Agent Group target type, the Name dropdown lists all Agent Group target types in your Configuration Database so you don't have to manually enter them. It also marks all used targets as taken by including these targets in the Check Integrity mechanism.

StrToLower

Parameters: String: STRING or variable (representing the string)

Return value type: STRING

This function returns a copy of the string with all the characters converted to lowercase. For example, as12FG is converted to as12fg.

StrToUpper

Parameters: String: STRING or variable (representing the string)

Return value type: STRING

This function returns a copy of the string with all the characters converted to uppercase. For example, as12FG is converted to AS12FG.

Target Manipulation Functions

Figure 244 shows the Function Properties dialog box with Target Manipulation selected.

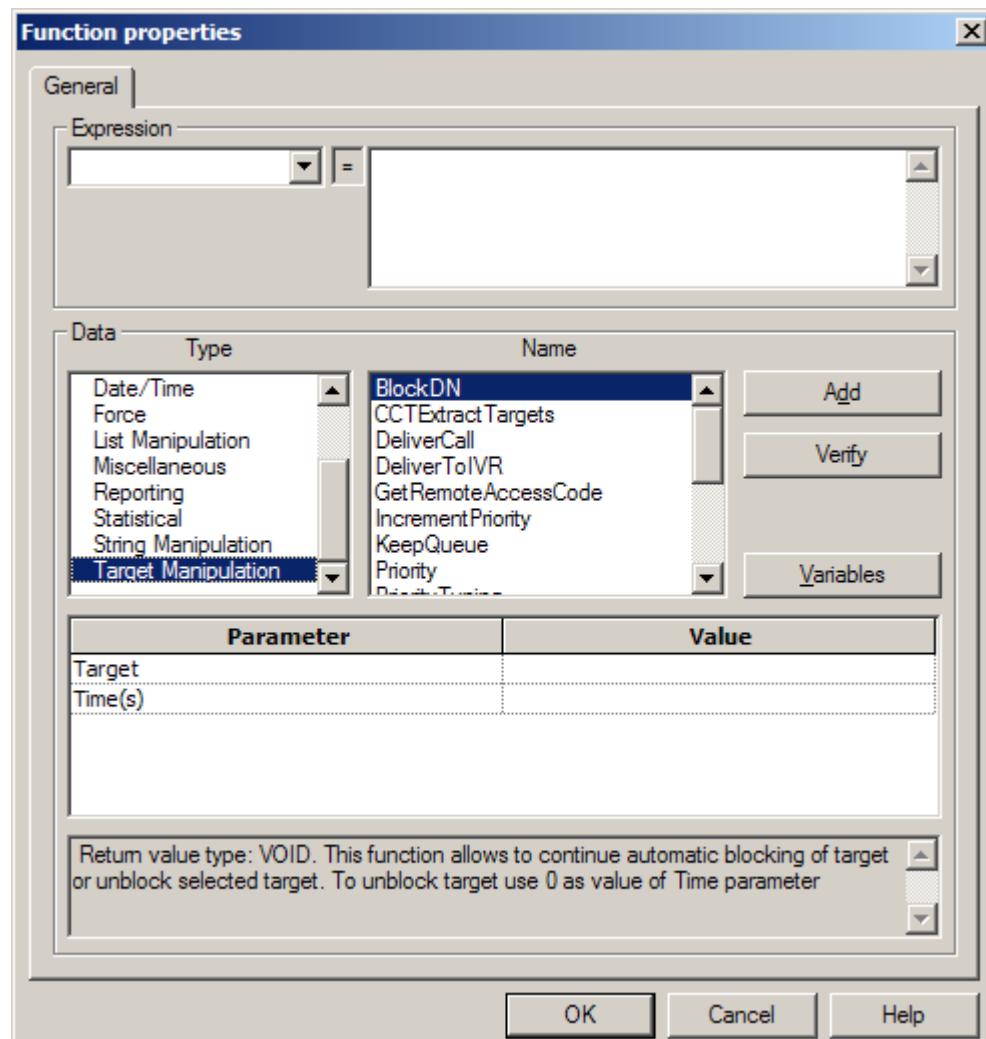


Figure 244: Function Properties Dialog Box, Target Manipulation Functions

This section presents functions in the Target Manipulation category (see Table 130 on [page 430](#)).

BlockDN

Parameters: Target: STRING (target) or variable (representing a string for the target DN).

Time: INTEGER representing the blocking time interval

Return value type: VOID

When functions SelectDN or SuspendforDN return a ready target, the target is always blocked by URS during some time in seconds (the value of `transition_time` option). The BlockDN function allows you override such behavior and change the time interval or even cancel it entirely (`Time` parameter = 0).

During the time the target is blocked, URS does not distribute any interaction to the target as specified by the first function parameter, which is the result of the SelectDN or SuspendForDN function.

This function can be used in cases where a target may have additional conditions (except a Stat Server-reported not-ready state) that should prevent the target from being selected as a valid target.

Note: Normally, the input for the BlockDN function is the output from the SelectDN or SuspendForDN function.

Example:

1. Assume the target is first selected as being ready from the Stat Server point of view.
2. As a result of being selected, the target will be blocked for the number of transition time seconds (blocking time interval).
3. Additional conditions can then be checked. For example, the remote target location trunk availability) can be checked.
4. If the target is not valid (there is no free trunk to this target, for example), the target should be unblocked to allow the target to be selected for other interactions.

DN Target Type

URS 7.0.1 introduced a target type called DN to support non-configurable targets. The target format for the Function Properties dialog box is `number@switch.DN`. Using this target is similar to forced routing, but with the following differences:

- More than one target can be used with random or percentage distribution (using a Percentage routing rule) or by Weight:

The weight assigned to the target in case of selection by percentage distribution; URS will attempt to route the interactions to the target according to the ratio between its weight and the sum of all weights in the target list. Format: Weight[number]DN@Switch.DN]

Example: Weight[1]1@Portland.DN

- URS can track target availability if T-Server provides the information.

The DN target is ready by default. In the case of EventError, URS considers the target not ready for a short time (15 seconds, by default).

CCTExtractTargets

Parameters: StatServer: STRING or variable (representing a string for a StatServer)

Data: STRING (list) or variable (representing a string)

Return value type: STRING

This function produces a list of targets from the supplied arguments and stores information associated to each target for subsequent use in number translation of type [TARGET.CCTN].

This function supports these target types:

- Agent
- Place
- Agent Group
- Place Group

The value of the StatServer parameter is used as the location attribute for each of the targets listed in the output.

The Data parameter can be a variable or a string. A string value must be a key-value list of the form:

```
TargetID1.TargetType1:Value1|...
|TargetIDN.TargetTypeN:ValueN
```

with no spaces in the list. The result of the function is a string of the form:

```
TargetID1@StatServer.TargetType1, ...
TargetIDN@StatServer.TargetTypeN
```

Thus, in the result targets are listed, separated by commas, and contain no spaces.

The values associated with each target are remembered and are accessible in Number Translation by means of the translation type [TARGET.CCTN].

Note: See Chapter 5 on [page 705](#) for a detailed explanation of Number Translation.

DeliverCall

Parameters: Destination: STRING (target) or variable (representing a string for the target)

Type: DELIVERBACK, DELIVERTOAGENT, DELIVERTOIVR, or variable

Return value type: STRING (list)

This function was designed primarily to shuttle interactions between IVRs and the original routing point. This function sends the current interaction to the prescribed target or back to the routing point where the interaction was first received if the Type parameter has a value of DELIVERBACK. The returned value can be `return:ok` or `return:error`.

Note: Normally, the input for the DeliverCall function is the output from the SelectDN or SuspendForDN function.

The operation of the function is described as follows:

1. If the destination parameter does not contain the string `return:ok`, then the function will immediately return `return:error`.
2. If the value of Type is DELIVERBACK, the Destination parameter is ignored (a string value must still be supplied for correct syntax). URS sends a routing instruction to T-Server to send the interaction to the routing point from where URS first received the interaction. The function returns `return:ok` after `EventRouteRequest` or `EventEstablished` is received for the interaction; that is, when the interaction arrives at any DN on the switch. The function returns `return:error` if T-Server sends `EventError` in response to the routing instruction sent by URS.
3. If the value of Type is DELIVERTOAGENT or DELIVERTOIVR, the Destination parameter provides information on the target to which the interaction will be sent. The DeliverCall function uses target information in the value formats returned by the functions SelectDN and Translate. Although a manually produced string can be used for the Destination parameter, it is strongly recommended to pass only values returned by one of these two functions.
4. If the state of the interaction does not allow T-Server to issue a routing instruction, then this function sets a corresponding error code and returns `Return:error`. If the target specified in the Destination parameter does not match to the last selected target saved in the interaction information structure, then URS issues a warning message but will continue executing the procedure.
5. If the Destination parameter contains a pair with the key `target_name`, then URS assumes that the target was previously translated and immediately proceeds to the next step. Otherwise, URS implicitly calls the function Translate and subsequently uses the key-value pairs of the translated target.

6. Finally, URS proceeds to issue a routing instruction using the key-value pairs in the target, after a possible translation. If the Type parameter is specified as DELIVERTOAGENT, URS waits to receive EventEstablished for the interaction from any point on the switch. If the Type parameter is specified as DELIVERTOIVR, URS waits to receive EventTreatmentRequired for the interaction from any point on the switch. When the corresponding event is received, the function returns return:ok. The function returns return:error if T-Server sends EventError in response to the routing instruction sent by URS.

Important Information

- The names DELIVERTOAGENT and DELIVERTOIVR are somewhat misleading. The choice between DELIVERTOAGENT and DELIVERTOIVR indicates which of the two events coming from the IVR port, either EventEstablished or EventTreatmentRequired, will be interpreted by URS as a signal that the function has finished its work. Whenever the function DeliverCall is used to send an interaction to a DN other than an IVR port, the Type parameter must have a value of 1 (DELIVERTOAGENT). When an interaction is sent to an IVR port, either DELIVERTOAGENT or DELIVERTOIVR can be used.

In the cases when return:error is returned by the DeliverCall function, the following error messages are possible:

- **0001 Unknown error**—for example, a failure to route; this message may be returned in case of an error different from any listed so far.
- **0002 Transfer in progress**—another transfer is in progress; this error message will be returned if URS has sent an instruction to T-Server to transfer the interaction between two DNs and if, at the time DeliverCall is invoked, no notification has been received by URS about the interaction reaching a DN after the transfer. This state can only be possible because treatments are running while the strategy executes the function.
- **0003 Treatment in progress**—a treatment is in progress; this error will occur if DeliverCall is invoked after URS has issued an instruction for T-Server to start a treatment but before T-Server responds with EventTreatmentApplied, EventTreatmentNotApplied, or an error.
- **0004 Call is on hold**—the agent has put the interaction on hold; this error message will be returned either if the interaction is put on hold at a DN for which URS is registered or if the interaction cannot be found on any of the DNs for which URS is registered.
- **0005 Call gone**—URS has not registered a DN belonging to the interaction.
- **0006 Operation impossible**—this error message will be returned if the requested transfer is impossible for technical reasons, for example,
 - if the strategy was started at EventRinging but EventEstablished has not been received from the DN,

- or if the interaction is to be transferred from an Electronic Audio Port to an Extension but the option `deliver_to_agent` is set to false.

In the latter case, use `DeliverCall['', DELIVERBACK]` first and then make the desired transfer.

- **0008 Routing done**—not permitted in postrouting; this message will be returned if the interaction has already been routed successfully by the `RouteCall` function or by a `Routing` object, or if the interaction has been force-routed either by the function `TRoute` or by combining the `Force` object with a `Routing` object.
- **0012 Bad target**—invalid target string; this error message will be returned if the key-value pair `result:ok` does not appear in the target as the first key-value pair.
- **0013 Remote error**—this error message will be returned if URS receives `EventError` from T-Server in response to the routing instruction.
- **0019 Translation failed**—likely due to incorrect configuration.

Diverting the Interaction from a Virtual Queue

When the parameter `Destination` contains a key-value pair with the key `vq` and the value corresponding to this key is a virtual queue on which the interaction has been placed, then the interaction will be diverted from that virtual queue after confirmation from T-Server that the interaction has left the DN. It will also be deleted from all internal router queues that are included in the virtual queue. If the virtual queue name is configured in Configuration Layer and belongs to the T-Server to which URS is connected, then `EventDiverted` will be generated for the virtual queue.

Note: The same list of error messages can be reported after a call of the function `RouteCall`.

DeliverToIVR

Parameters: `Target`: STRING (statistical object) or variable (representing the string for the target)

`Timeout`: INTEGER or variable

Return value type: VOID

The purpose of this function is to explicitly send an interaction to an IVR, but unlike `DeliverCall`, the function `DeliverToIVR` receives as a parameter a target specified in a high-level target format, either `Name@StatServerName.Type` or `DN@Switchname`. After the function is executed, the current interaction is still controlled by URS and the strategy continues its execution.

In IRD, the target types in the dialog box for this function include:

- `Destination Label`

- Place
- Place Group
- Queue
- Queue Group
- Routing Point (virtual and real)
- Variable

For all target types but Variable, you also specify the name of the target for the selected type and its location (Stat Server name). For the Variable target type, select the name of the variable only.

The conventions of the target specification are the same as for specifying an IVR as a busy treatment. In particular, if Type is absent from the string defining the target, URS interprets it as a target for forced routing and immediately instructs T-Server to send the interaction to the specified DN at the specified switch. Otherwise, URS inquires from Stat Server for target availability.

If the target type is Group of Places (.GP), it is available when at least one of the places is available. If the target is available, URS instructs T-Server to send the interaction to a ready DN reported by Stat Server along with other information on the target. If the target is not available, URS waits for it to become available. If it does not become available during the specified timeout interval, the function causes an error in the strategy. You can treat this case by using the Error Segmentation object (see “Error Segmentation” on [page 123](#)).

Important Information

- See “DN Target Type” on [page 589](#) for more information on DNs as targets.
- Although the function DeliverToIVR was designed to send calls to an IVR, it doesn’t check whether the specified target is an IVR or not. Therefore, you can choose the best IVR configuration for your needs, such as an individual place or group with an associated Electronic Audio Port, a queue from which calls are distributed to available IVRs, and so on.

The following error messages are possible:

- -001 Timeout
- 0001 Unknown error—for example, invalid target or fail to route call
- 0008 Routing done—not permitted in postrouting
- 0013 Remote error—T-Server error

GetRemoteAccessCode

Parameter: Destination: STRING (target) or variable (representing a string for the destination)

Return value type: STRING (target)

This function instructs URS to inquire from a remote T-Server for an access telephone number for the purpose of external routing (for additional information, refer to Multi-Site Support in any Genesys T-Server deployment guide).

The `Destination` parameter is a key-value list that must contain the keys `dn` and `switch`. In practice, the `Destination` parameter should have for a value the result of one of the following functions: `SelectDN`, `Translate`, or `SuspendForDN`.

The function returns a key-value list where all the keys have the same values as in `Destination`, with the possible exception of the keys `return` and `dn`.

Important Information

- Normally URS itself performs all external routing operations if you configure the corresponding Application options. Use this function to perform external routing independently from that specified in the URS Application object Options tab.
- (Network Routing only) Network T-Server (versions 6.1 and later) provides the Access Number to URS for the remote T-Server. Remote, premise T-Servers connected to Network T-Servers of version 6.1 or later, no longer have to be connected to URS directly unless agent reservation is required.
- External routing for T-Servers is called Inter Server Call Control (ISCC). The Genesys T-Server documentation also uses the term “client-designated” external routing. Generally, for ISCC transactions, T-Server asks another site (T-Server) for the switch’s access number using `TGetAccessNumber`, then transfers a call to that site. With client-designated external routing, URS (as a T-Server client) sends `TGetAccessNumber` to acquire the necessary info. When received, URS sends this information to an appropriate T-Server, which transfers the call to the new destination. This client-designated (or URS-controlled) feature is used when T-Servers are not talking/configured with each other or if it is more convenient that way for a particular site.

The key `return` receives `error` as a value in the following cases:

- `0001 Unknown error`—for example, a failure to send `GetAccessNumber` request
- `0002 Transfer in progress`
- `0003 Treatment in progress`
- `0004 Call is on hold`—the agent has put the interaction on hold
- `0005 Call gone`—URS has not registered a DN belonging to the interaction
- `0006 Operation impossible`—the interaction cannot be routed from the current DN
- `0008 Routing done`—not permitted in post routing
- `0012 Bad target`—invalid target string

- `0013 Remote error`—T-Server error
- `0015 Closed server`—no connection with T-Server
- `0018 Unknown object`—unable to find switch or T-Server for the target

If the function does not return any errors, the key `return` receives `ok` as a value. If the value in `Destination` corresponding to the key `switch` is the name of the switch where the interaction is currently located, no further processing is done, and the key `dn` retains its value from `Destination`. If the specified switch is different from the current switch, URS uses the external routing protocol to inquire for an access number from the remote T-Server, and the obtained number is provided in the result as the value corresponding to the key `dn`.

IncrementPriority

Note: This function replaces the `Increment` function, which has been retired.

Parameters: `Increment`: INTEGER (delta of priority, can be positive or negative)
`Interval`: INTEGER (time interval priority is incremented)

Return value type: VOID

This function is extended version of the `Increment` function and results in incrementing interaction priority by the `Increment` every `Interval` second. This function affects the priority of an interaction for all targets: targets the interaction is already waiting for and those it may be waiting for in the future.

Note: `Interval` can not be less than 5 seconds (URS will always use as the `Interval` the value `max(Interval, 5)`).

IncrementPriorityEx

Parameters: `Increment`: INTEGER (delta of priority, can be positive or negative)
`Interval`: INTEGER (time interval priority is incremented)
`Initial_Interval`: INTEGER (time interval to wait before starting to increment)

Return value type: VOID

Compared to the existing `IncrementPriority` function, this function provides the capability to increment priority starting *after* a specified time interval. Users can instruct URS to begin to increment priority in `N` seconds after this function is called, and thereafter increment the priority by `x` number every `y` seconds. For example: “in one hour, begin to increment priority by 10 every minute.” This corresponds to the following specific function:

`IncrementPriorityEx(10, 60, 3600)`.

KeepQueue

Parameter: KeepQueue: CONSTANT or variable

Return value type: VOID

This function prevents URS from distributing any interaction with a priority less than current one if the function is set to true. The default value for this function is `false`.

When examining interactions within a queue waiting for a target, URS searches for the interactions with the maximum priority and stops when it finds interactions that can be routed or encounters an interaction with `KeepQueue[true]`. If an interaction with `KeepQueue[true]` cannot be routed immediately, URS must maintain its priority in queue and cannot pass over it to route a low-priority interaction. Until this interaction is routed, no other interactions with a lower priority are routed to the selected target.

Example KeepQueue Usage:

- A strategy uses a busy treatment on an IVR that URS does not control.
- An interaction arrives at a routing point and is sent to the IVR because the agent is busy.
- Then the agent becomes available but URS cannot send the interaction to the agent because URS does not control the IVR where the interaction is located.
- Another interaction arrives.
- Without `KeepQueue` set to `true` for the first interaction, this second interaction would be sent to the agent.
- With `KeepQueue` set to `true` for the first interaction, the second interaction is sent to another IVR port. The second interaction cannot be sent to the same IVR port as the first interaction because this IVR port is busy with the first interaction.
- When the first interaction returns from the IVR, the interaction is routed to the agent.
- If the first interaction never returns from the IVR (for example, if the interaction was abandoned) and the URS option `call_tracking` is set to `true`, URS will delete this first interaction after a minute to unblock access to the agent.

Warning! To avoid the situation where one interaction with `KeepQueue[true]` prevents all subsequent interactions from ever being routed, do not activate `KeepQueue` for a long time (set it to `true` in one place in a strategy after specifying a wait time for a target, and then set it to `false`) or set the URS option, `call_tracking` to `true`. By setting `call_tracking` to `true`, URS will remove any interactions that have not moved off the routing point after one minute.

Use this function when adherence to call priority is critical.

This function returns no errors.

Important Information

- Using this function could cause longer waiting time for interactions in queues and so affect the reporting statistic for queues due to the adherence to call priority.
- Busy treatment settings for `transition_time`, `transfer_time`, and `timeout` and the setting of the KeepQueue function do not affect one another.

NMTEExtractTargets

Parameters: `Data`: string (a comma-separated list of elements representing a targets list where the `RecordSize` value describes one target)

`RecordSize`: INTEGER representing the number of items used to describe each target in the targets list returned from the query (valid values are 1-4). Each target can be described using four elements: `Data`, `Threshold`, `Speaking Time`, and `RetryTime`. Note: IRD verifies only the type of entered value (must be Integer), but not boundaries.

`Default Threshold`: INTEGER

`Default Speaking Time`: INTEGER (if you do not specify a value, the default speaking time is set to 0)

`Default Retry Time`: INTEGER (if you do not specify a value, the default retry time is set to 0)

Return value type: STRING comma separated list of targets with thresholds (see “[SetThresholdEx](#)” on [page 577](#)) based on the `RStatCallsInQueue` statistic.

This function enables you to track the number of active calls at a DN that is not configured in the Configuration Database. It establishes a counter for all active calls at the non-configured DN. Using this counter, you can compare the number of active calls to a specified threshold, and stop routing calls to the DN when the threshold has been reached.

The NMTEExtractTargets function parses a targets list, which is a comma-separated list of elements produced by a database query. Each target can be described using four elements: `Data`, `Threshold`, `Speaking Time`, and `RetryTime`.

These parameters are explained below:

- `Data`—Identifies the target.
- `Records size`—How many elements are entered to describe each target. If some of four elements listed in the following bullets are absent (only `Data` is mandatory) the corresponding function parameters are used instead.

- **Threshold**—The maximum number of calls on the destination when this destination is considered to be ready. The threshold value is considered as attribute to a call. NMTExtractTargets sets this attribute for each call it evaluates.
- When it is time for the call to be routed to the non-configured DN, URS compares the call's threshold attribute to the counter. If the counter shows a value greater than the call's threshold attribute, the call is not routed until the counter's value drops below that threshold.
- **Speaking Time**—URS considers a call to a non-configured DN to be terminated when T-Server reports that it is terminated. However, if no information is received from T-Server in <MaxCallLiveTime> seconds, URS considers the call to be terminated.
 - **Retry Time**—If URS receives an error message in response to a routing request to a non-configured DN that indicates that the number of calls at the destination is different from the number URS believes are there, URS temporarily stops sending calls to that DN for <Retry Time> seconds. This delay enables synchronization of the number of calls on the DN with the number that URS believes are there.

Examples

The following is an example of input and output for the NMTExtractTargets function:

```
NMTExtractTargets['1111111, 22222, 333333', 1, 10, 120, 20] =
'{RStatCallsInQueue<=10}1111111.DN,
{RStatCallsInQueue<=10}22222.DN, {RStatCallsInQueue<=10}333333.DN'
```

Where '1111111, 22222, 333333' is the targets list Data, 1 is the number of items per target (Record size), and 10, 120, and 20 are Default Threshold, Default Speaking Time, and Default Retry Time respectively.

Another sample:

```
NMTExtractTargets['1111111, 10, 22222, 20, 333333, 30', 2, 0, 120, 20] =
'{RStatCallsInQueue<=10}1111111.DN,
{RStatCallsInQueue<=20}22222.DN, {RStatCallsInQueue<=30}333333.DN'
```

Using the Function

Use of this function to track the number of calls at non-configured DNs usually includes:

1. Setting a threshold, using the SetThresholdEx function, for every non-configured DN that you want to include. The only threshold you can set for non-configured DNs applies to the value returned by the RStatCallsInQueue statistic.

Alternatively, you can set a threshold by prefixing `Threshold{Statistic op value}` before the target specification. When you set a threshold in this

way, the NMExtractTargets function automatically augments the targets with prefixes in the indicated format. The output of this function is a comma-separated list of targets, with thresholds, that is ready to be used as a parameter of standard target-selecting functions or objects.

2. Configuring URS to count the `RStatCallsInQueue` statistic for non-configured DNs.
3. Setting all the non-configured DNs as targets in Selection object (see [page 326](#)).

Note: URS has no information about the current state of the DN—only the number of calls active on it. Active calls are considered to be the number of calls URS sent to the target minus the number of calls that were terminated.

In addition to generating output, the NMExtractTargets function also overwrites the default values (15 seconds and 600 seconds) for RetryTime and Speaking time. Thresholds for non-configured DNs set using the SetThresholdEx function, for example, use the current default values for RetryTime and Speaking Time that were set by the latest NMExtractTargets function.

Note: Changes applied to default settings (RetryTime, Speaking Time) specified in the NMExtractTargets function work only for non-configured DNs that were created after NMExtractTarget function was used. Those non-configured DNs that were created previously are not affected by any changes made using the NMExtractTargets function.

If the information about the non-configured DNs (numbers, threshold values) is retrieved from a database, use the NMExtractTargets function to transform the result of the database query into a target list suitable for use in the Selection object (rather than doing it manually). In this case, NMExtractTargets automatically sets all the specified thresholds and you do not need to call the SetThresholdEx function.

Setting Multiple Thresholds

You can set more than one threshold by using the NMExtractTargets function in more than one strategy or using it more than once in a single strategy. In this way you can set a different allowable number of concurrent calls for different situations, as described in “Using NMExtractTargets to Set Multiple Thresholds—Example” on [page 601](#).

Note: If you use the same non-configured DN in other strategies or in multiple places in the same strategy, the same counter is used for that DN.

Using NMTEExtractTargets to Set Multiple Thresholds—Example

You can set different thresholds for the same non-configured DN to take account of different business conditions.

For example, between 9:00 AM and 5:00 PM, an outsourcer with a non-configured DN can handle 100 concurrent calls. After 5:00PM, the outsourcer can handle only 50 concurrent calls on this DN.

Here is how it works:

1. Place the NMTEExtractTargets function in two places. You can use it twice in one strategy or use two separate strategies to handle this situation, depending on what is best suited to your environment. A global counter is created for the non-configured DN.
2. Using one instance of the function, set a default threshold value of 100 to be attached as an attribute to all calls that arrive between 9:00 AM to 5:00 PM.
3. Using the other instance of the function, set a default threshold value of 50 to be attached as an attribute to all calls that arrive between 5:00 PM and 9:00 AM.

When a call is ready to be routed, the call attribute is compared to the global counter, which is incremented or decremented to correspond to the number of active calls on this non-configured DN. If the value on the counter is greater than the call's Threshold attribute, the call is not routed until the counter's value drops below the appropriate threshold.

Deployment Considerations

Only one URS can send calls to the non-monitored DN. If you require multiple Service Numbers to distribute to the same non-monitored DN, align the SN-to-DN table, the network switch, and URS so that all calls sent to a non-monitored DN are processed by the same URS.

You can use a Network T-Server or a SIP Server as the network switch.

To use this function, URS must register on a special DN, called `switch::` in order to receive notification from T-Server about the termination of calls.

If you do not register, the `RStatCallsInQueue` statistic is not correctly calculated for non-monitored targets. Registration is not automatic and is controlled by URS's “[call_monitoring](#)” option (see [page 634](#)).

Priority

Note: For additional information, see the business-priority routing section in the *Universal Routing 8.0 Routing Application Configuration Guide*, Recommended Settings table.

Parameter: Priority: INTEGER or variable

Return value type: VOID

This function sets the default priority for the current interaction (sets the value to be used when the interaction is placed in the queue). For example, if an incoming interaction is identified as preferred (by its ANI or the IVR), it can be routed ahead of those that are already waiting. To increase the priority as time goes by, use the IncrementPriority function (see [page 596](#)). The default priority of the interaction is used when an interaction is placed in the internal router queue of a target list.

You can use the Priority function multiple times before a target Selection object to affect the content of the targets within that target Selection object only, such as during recursive (cascading) targeting. [Figure 245](#) shows an example strategy where the Priority function is used multiple times in a strategy.

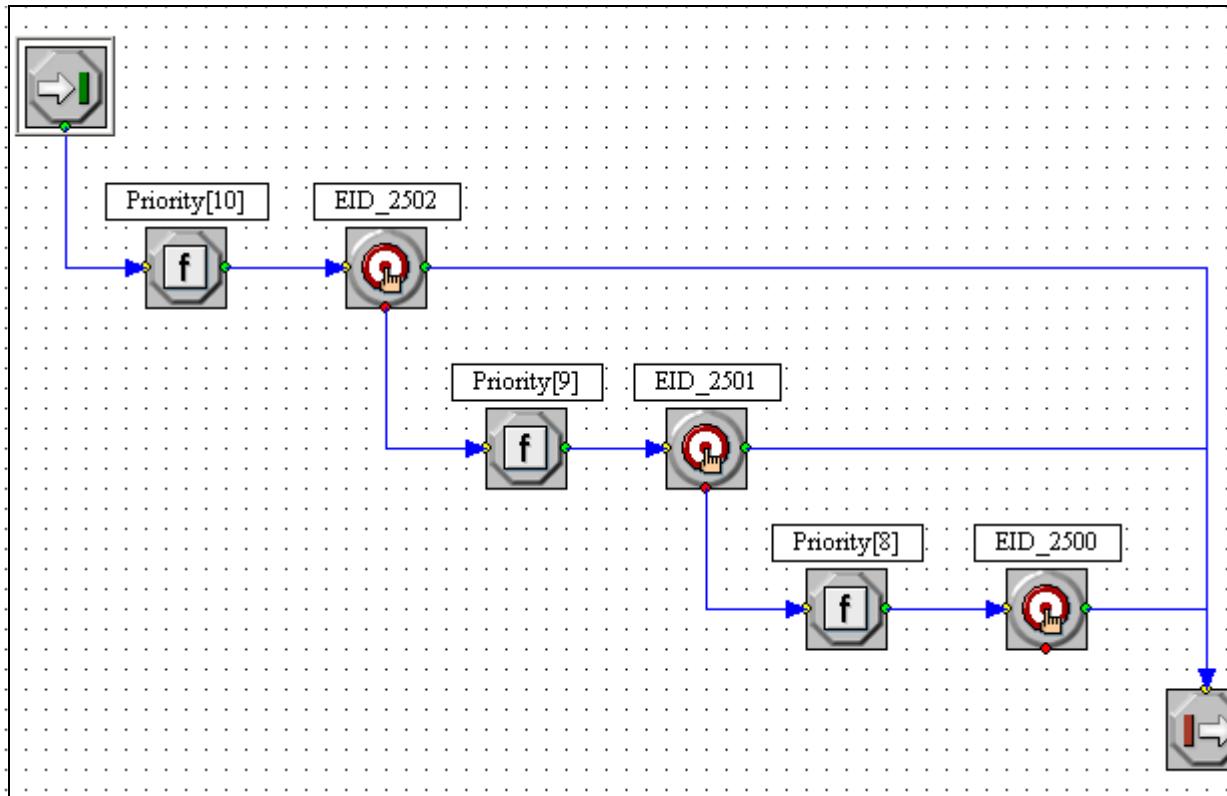


Figure 245: Example Strategy Using Priority Function

In the example strategy in [Figure 245](#), if an interaction with the priority set to 10 is not routed in the first target Selection object, the interaction goes out the red port to the next Function object where the priority is decreased to 9. If the interaction is still not routed in the second target Selection object, the priority is decreased again to 8 and goes to the third target Selection object.

Important Information

- The Priority function does not affect the priority of the interactions that were waiting for a target. To do this, use the SetVQPriority function (see [page 613](#)).
- The Priority function does not affect the specific internal router queue priorities set by the functions SelectDN (see [page 608](#)) and SetVQPriority (see [page 613](#)).
- Greater numbers correspond to higher priorities. The default priority of an incoming interaction is zero.

PriorityLimits

Note: For additional information, see the section on business-priority routing in the *Universal Routing 8.0 Routing Application Configuration Guide*. Also note that Genesys currently does not support “What-If” wait time business-priority routing for non-voice interactions processed outside of URS.

Parameters: Lower Limit: INTEGER
Upper Limit: INTEGER

Return value type: VOID

This function enables users to define the upper and lower limits of a priority value. The priority of interactions with this function applied cannot, under any circumstances, exceed the value of Upper Limit or be lower than the value of Lower Limit. Previously, priority limits for any call were hard-coded and ranged from -100000000 to 100000000, which remains the default range for this function unless Lower Limit and Upper Limit are specified.

PriorityTuning

Note: For additional information, see the section on business-priority routing in the *Universal Routing 8.0 Routing Application Configuration Guide*. Also note that Genesys currently does not support “What-If” wait time business-priority routing for non-voice interactions processed outside of URS.

Parameters: UseAgeOfInteraction: INTEGER (true or false)

UsePrediction: INTEGER (true or false)

Return value type: VOID

URS always puts interactions into waiting queues according to their priorities. The PriorityTuning function defines how URS handles interactions with the same priorities. By default, interactions with the same priority are ordered according to the time the interaction began to wait for some target.

UseAgeOfInteraction Parameter

If parameter `UseAgeOfInteraction` is true, URS uses the time the interaction was created instead of the time interaction is placed into the waiting queue. Age of interaction is usually the time that URS starts the strategy for the interaction.

Setting the interaction age enables you to safely exit and reenter routing objects. The interaction does not lose its position in queue, because its position is based on the age-of-interaction value, which is not affected by movement from object to object.

Strategy Setup: `UseAgeOfInteraction = true`

Set up the strategy as follows:

1. Use function `SetInteractionAge(Keep)` to timestamp the interaction. Age of interaction will be counted from this moment on.
2. Use function `PriorityTuning` to prioritize this interaction among others in the queue according to their age. Set parameter `UseAgeOfInteraction` to true.

[Figure 246](#) shows how this looks in a strategy.

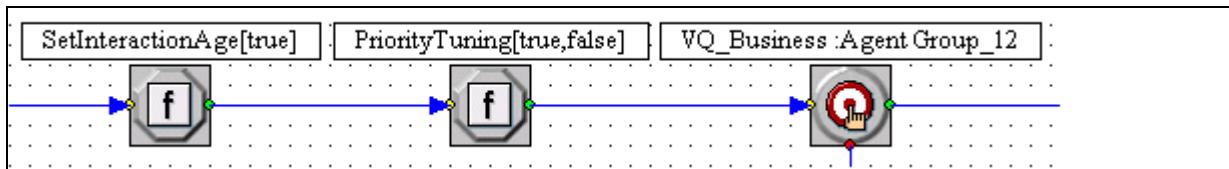


Figure 246: PriorityTuning with `UseAgeOfInteraction` Set to True

Use case:

- Interactions queued by FIFO.
- Business process and call flow requires many transfers/collaborations.
- The age of interaction needs to be known to adjust interaction priority based on age.

UsePrediction Parameter

If `UsePrediction` is `true`, then URS calculates the estimated time for the interaction to be answered and will use this time instead of the time that the interaction has already waited.

Strategy Setup: UsePrediction = true

Set up the strategy as follows:

1. Use Function PriorityTuning to adjust interaction priority based on what-if wait time
2. Set parameter `UsePrediction` to `true`.

[Figure 247](#) shows how this looks in a strategy.

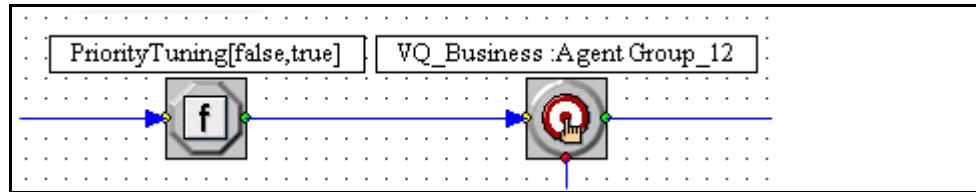


Figure 247: PriorityTuning with UsePrediction Set to True

Use case:

- Interactions queued by FIFO.
- Contact center staffs queues unevenly and agents are shared among these queues.

For example, if interaction #1 waits 10 seconds and interaction #2 waits 20 seconds, then URS puts interaction #2 in front of interaction #1 in the waiting queue ($20 > 10$). If, however, `UsePrediction` is `true`, URS calculates the estimated time for the interaction to be answered. In this case, URS might estimate the waiting time for interaction #1 as 60 seconds and the waiting time for interaction #2 as 40 seconds. In this case, URS puts interaction #1 in front of interaction #2 ($10+60 > 20+40$) in the queue.

PriorityTuning with SetInteractionAge and use_service_objective

By default, interactions with the same priority are ordered according to the time the interaction began to wait for some target. The `PriorityTuning` function., together with `SetInteractionAge` function and the `use_service_objective` option, changes the default behavior as described below.

PriorityTuning and SetInteractionAge Function

The `SetInteractionAge` function (see [page 545](#)) can modify interaction age as follows:

- Set parameter `UseAgeOfInteraction` of function `PriorityTuning` to `true`.
- Set function `SetInteractionAge` to `true` in the strategy loaded at the very first routing point the interaction enters.

Use the above when the business scenario exists:

- The contact center has lots of interactions transferred/collaborated due to business process needs.
- The *relative age* of an interaction should be known so that interaction priority in a waiting queue can be adjusted based on the age. (Relative age is the time accumulated since interaction entered very first routing point.)

Note: The time that the strategy was started and the time the interaction was queued can be very different if mandatory treatments are used at the head of the strategy before target selection.

See the *Universal Routing 8.0 Routing Application Configuration Guide* for information on using the `PriorityTuning` and `SetInteractionAge` functions for handling ring-no-answer scenarios.

PriorityTuning and the `use_service_objective` Option

The complementary `use_service_objective` option (see [page 681](#)) allows you to scale the time interval for different interactions. If this option set to `true`, URS scales the time interval that the interaction waits (or is going to wait) according to the Service Objective of the interaction (see “MultiAttach” on [page 140](#) for information on Service Objective).

If two interactions wait the same time, such as 60 seconds, but interaction #1 has a Service Objective of 30 seconds and interaction #2 has a Service Objective of 120 seconds, then URS puts interaction #1 ahead of interaction #2 in the queue ($60/30 > 60/120$).

Note: The use of Service Objective for priority tuning is on the server level. You cannot use it for specific routing points or queues (virtual queues).

Warning! The interaction selection criteria associated with the Priority Tuning function (age of interaction, relative wait time (such as wait time in queue or predictive wait time), service objective risk factor, or any combination of these parameters) are not supported in a multi-URS environment in the scenario where the same target is selected from multiple interactions by different URSSs.

RouteCall

Parameter: Destination: STRING (target) or variable (representing a string for the destination)

Return value type: STRING (target)

This function attempts to definitively route the current interaction to a target specified in low-level format. Its operation is very similar to that of the function DeliverCall with Type parameter equal to 1 (`DeliverToAgent`) or 2 (`DeliverToIVR`). The returned value also obeys the rules listed there.

This function differs from DeliverCall in the following ways:

1. The function RouteCall waits for `EventRouteUsed` from T-Server in order to conclude that the interaction has been sent to the destination successfully. Thus, it is not concerned with the actual arrival of the interaction at the destination DN.
2. After the current interaction has been routed successfully, URS no longer controls it. After the definitive routing of the interaction, URS proceeds to execute the postrouting actions, specified in the sequence of objects.

Note: Usually the input for this function is the output from the `SelectDN`, `SuspendForDN`, or `Translate` function. You also can manually construct the target string, but in this case URS performs routing in force mode. That is, no statistics calculated by URS are updated.

Important Information

- The interaction is diverted from all virtual queues and all internal router queues after confirmation only. `EventDiverted` is generated for all configured virtual queues on which the interaction has been waiting. When the function returns with a Remote error message, the interaction is not removed from virtual queues or internal router queues.
- The possible error messages are the same as those for DeliverCall. See "DeliverCall" on [page 591](#).
- URS does not correctly calculate the percentage allocation if functions `SelectDN` and `RouteCall` are not in the same strategy.

Routed

Parameters: none

Return value type: none

This function marks an interaction as routed. When routing functions, such as `RouteCall`, are successfully executed, an interaction is implicitly marked as routed. This function explicitly marks the interaction as routed.

This is important when successful routing could simply be applying a treatment or sending an automated e-mail response. Since no routing object is being executed, URS has no information about which action means that the interaction was successfully routed. This function allows URS to dispose of the interaction when the strategy is completed and not route the interaction to the default destination.

SelectDN

Parameters: `Virtual Queue`: STRING or variable (representing a string for the virtual queue)

`Priority`: INTEGER or variable

`Statistic`: STRING or variable (representing a string for the statistic)

`Selection Flag`: `StatSelectMax`, `StatSelectMin`, or `StatSelectAny`

`Target`: STRING or variable (representing a comma-separated list of targets: Agent, Agent Group, Campaign, Campaign Group, Destination Label, Place, Place Group, Queue, Queue Group, Routing Point)

Return value type: STRING (target)

This function corresponds to the initial steps of the operation of a Routing object ([page 311](#)):

- creating an internal router queue as part of a prescribed virtual queue,
- inquiring from Stat Server on the availability of any of the listed targets and,
- if any of the targets are available, choosing one according to some criterion.

If no available target is found, the function returns `return:timeout`; otherwise, the returned result is a key-value list containing the pair `return:ok` and a low-level target specification consisting of detailed information on the chosen target, obtained from Stat Server.

Note: The output of this function is usually used as input for other functions, such as `BlockDN`, `Translate`, `DeliverCall`, or `RouteCall`.

List Returned by SelectDN

If the function `SelectDN` succeeds in selecting an available target, then, in addition to the pair `return:ok`, the list returned by the function will contain the following keys:

`target_name`—the name of the selected target from the list.

`target_location`—the location of the selected target from the list.
`target_type`—the type of the selected target from the list.
`vq`—the virtual queue name specified as a parameter of the function `SelectDN`.
`dn`—an available DN of the selected target; this value is reported by Stat Server. If the target is an agent and the option `use_agentid` is set to true, then the value of this key will not be a DN but the Employee ID of the agent.
`switch`—the switch where the DN provided for the selected target is located; this value is also reported by Stat Server.
`agent`—the Employee ID of the agent selected; this value, reported by Stat Server, is present only if the target is of type Agent (.A) or Group of Agents (.GA).
`place`—the name of the place selected; this value, reported by Stat Server, is present only if the target is of type Agent (.A), Group of Agents (.GA), Agent Place (.AP), Group of Places (.GP) or Campaign Group.

Virtual Queue Parameter

The parameter `Virtual Queue` can contain an arbitrary string or variable. If the empty string is passed, the interaction is placed on the generic virtual queue. If the virtual queue name given in this parameter is the same as the Alias of a virtual queue configured in the Configuration Database, and if the interaction is not already waiting on that virtual queue, `EventQueued` is sent; the parameter is used by Stat Server to maintain virtual queue statistics.

Priority Parameter

The `Priority` parameter sets a priority of the interaction for the newly created internal router queue. If you want to use the global priority of the interaction for the internal router queue, you should pass to this parameter the value returned by the function `GetPriority`. This feature of the function `SelectDN` has no analog in a Routing object.

Note: The priority of an interaction assigned by the function `SelectDN` is specific to the internal router queue, and does not affect the global priority of the interaction. After a specific internal router queue priority is assigned, it remains in effect until the next time it is changed by the function `SetVQPriority` or increased by the action of the latest `Increment` object.

Statistic Parameter

The `Statistic` parameter only affects the operation of the function if the `Selection Flag` parameter equals `StatSelectMax` or `StatSelectMin`. In this case,

the value of this parameter must be a predefined statistic or a statistic defined within the strategy.

Target Parameter

In IRD, the target types in the dialog box include:

- Agent
- Agent Group
- Destination Label
- Place
- Place Group
- Queue
- Queue Group
- Routing Point (virtual and real)
- Variable

For all types but Variable, you also specify the name of the object for the selected type and its location (the Stat Server name). For the Variable type, select the name of the variable only.

The Target parameter can be a string or a variable to specify a target parameter. You can specify one or more targets or create a variable and using the Assign function, assign a list of targets to that variable.

The value of every target parameter (as opposed to a variable parameter) must be a comma-separated list of syntactically correct high-level targets (Name@StatServerName.Type when written in full notation). The targets can be in reduced notation, with StatServerName or Type omitted. If StatServerName is omitted, the name specified in the configuration option default_stat_server is used; the value used for an omitted Type is obtained from the configuration option default_object. These rules for completing targets specified in reduced notation are the same as for a routing object.

Different weights can be assigned to different targets. If you use weights, make sure that all the specified targets on all the lists receive weight numbers. URS parses out the weight number by searching for the first instance of square brackets [], between which an integer value must be specified. Thus, the initial substring leading to the opening square bracket is ignored by URS. Two intuitively appealing choices for the format are as follows:

- Immediately specifying a weight in square brackets, followed by a high-level target, for example, [40]AgentX@Reconnaissance.A.
- Using the word *Weight* in front of the number in square brackets for the sake of a visual analogy with a SELECTION statement, for example, Weight[40]AgentX@Reconnaissance.A.

Be aware, however, that URS interprets `WildStrawberries[40]AgentX@Reconnaissance.A` in the same way as either of the above two formats.

When weights are assigned to targets, URS selects targets for which the ratio of the target weight to the sum of all weights from all lists is greater than the ratio of the number of calls routed from the internal router queue to that target to the total number of calls routed from the internal router queue.

If weights are assigned to targets, any specified statistic is ignored regardless of the value of the parameter `Selection Flag`.

As an alternative to using the `SetThresholdEx` function, you can specify a threshold for each target by prefixing the target specification with `<Threshold>{<statistic> op <value>}`.

For example, entering `Threshold{StatCallInQueue<5}Queue.Statserver.Q` has the same result calling the function `SetThresholdEx[Queue.Statserver.Q, StatCallsInQueue, 5 ReadyIfLess]` with the subsequent targeting of `Queue.Statserver.Q`.

Important Information

- URS does not correctly calculate the percentage allocation if functions `SelectDN` and `RouteCall` are not in the same strategy.
- See “DN Target Type” on [page 589](#) for more information on DNs as targets.
- If a target is assigned a weight of 0, it can only be selected if no target of positive weight is reported as available by Stat Server.
- Spaces are ignored between the square brackets designating the weight of a target. If any character other than a space or a decimal digit occurs between the square brackets, the corresponding target is assigned a weight of 1.
- The `BlockDN` function (see [page 589](#)) blocks and unblocks a DN for a specified period of time.

The following errors are possible:

- `-001 Timeout`—the reservation or treatment delay time is over or there are no targets.
- `0001 Unknown error`—for example, invalid target selection.
- `0008 Routing done`—not permitted in postrouting.

Specifying the Targets in Target Lists

All the possible ways for specifying targets in a `Routing` object are also available in the case of the function `SelectDN`. Therefore, for the `TargetList` parameters, you can freely use explicit string constants, variables, functions, and compositions of functions, including the special function `UData` and the function `Skill`.

SetTargetThreshold

Use this function to define the statistical threshold for imposing additional readiness conditions for targets. One possible use for this function is in the Multi-Function object when implementing a Share Agent by Service Level Agreement routing solution. In this case, it can define the statistical thresholds for borrowing and lending agents.

Note: Threshold expressions support all arithmetic operations: +, -, *, /. Threshold expressions are used as parameters for the functions `SetTargetThreshold` or as the Threshold value in the Target Selection object.

Parameters: `Target`: STRING (statistical object) or variable (representing a string for the target that the routing condition is imposed upon, such as a string for the target that can lend agents if all conditions are met such as shown in Figure 82 on [page 148](#)). Similar to the existing `SetThresholdEx` function on [page 577](#).
`Statistical Expression`: STRING. Represents statistical (threshold) expression representing a condition that must be true for the conditional routing to occur, such as the borrowing/lending expression shown in Figure 82 on [page 148](#).

[Figure 248](#) shows a strategy that uses the Multi-Function object and `SetTargetThreshold`.

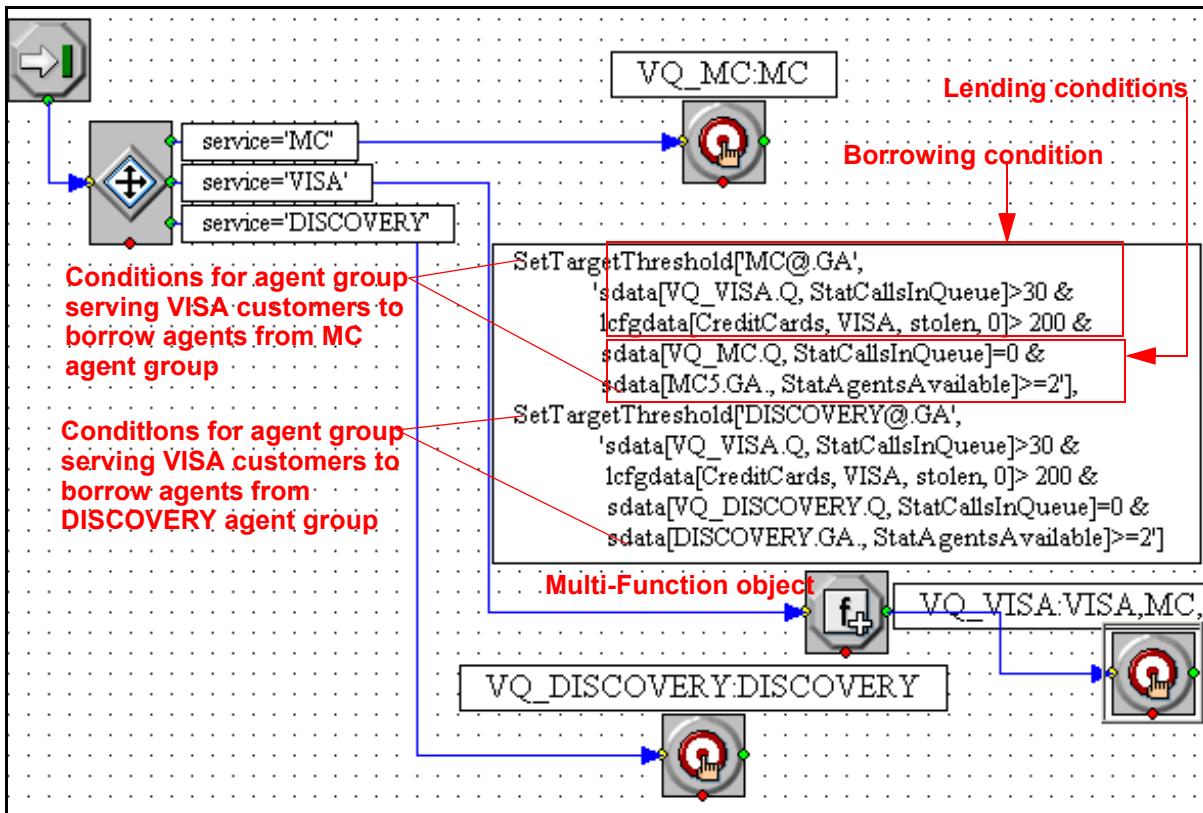


Figure 248: SetTargetThreshold Specified in Multi-Function Object

Refer back to Figure 81 on [page 147](#) to see the properties dialog box associated with the MultiFunction object. The MultiFunction object properties dialog box reflects what was previously selected in the Function Properties dialog box (see Figure 82 on [page 148](#)).

For detailed information on using the SetTargetThreshold function, see the chapter on Share Agent by Service Level Agreement Routing in the *Universal Routing 8.0 Routing Application Configuration Guide*.

Also see “DeliverCall” on [page 591](#).

SetVQPriority

Note: For additional information, see the section on business-priority routing in the *Universal Routing 8.0 Routing Application Configuration Guide*.

Parameters: VQ Name: STRING or variable (representing a string for the VQ Name)

Priority: INTEGER or variable

Return value type: VOID

This function explicitly sets the virtual queue priority of the current interaction for a specified virtual queue. This priority will become the internal URS queue priority for every internal URS queue composing the virtual queue, and it invalidates the priorities set for these internal URS queues. Every time the specific internal URS queue priority of an interaction is reset, it is moved ahead of all lower-priority calls and behind all higher-priority calls and previous calls with the same priority in the URS waiting queue.

When defining `VQ Name`, the value can be specified in two ways:

- Enter the string manually.
- From the dialog box, opposite `VQ Name`, click under `Value`. In the resulting `Key` dialog box, select a Type (`VQ Name` or `Variable`), and then select or enter a value. The values available are for the configured `VQ Name`. The values available for the `Variable` type are the variables included in the `Variable` list.

When the function is called with the parameter `VQ Name` set to the empty string, all internal URS queue priorities for the current interaction are reset, but the function `SetVQPriority` still affects the global priority of the interaction.

SuspendForDN

Parameter: `Timeout`: INTEGER or variable

Return value type: STRING (target)

This function waits for an available target from any of the virtual queues on which the current interaction has been placed. The wait is a length of time specified in seconds by the `Timeout` parameter.

If the timeout period elapses without a target becoming available, the function returns `return:timeout`. For example, a target does not become available to the current interaction if an earlier interaction with the same or higher priority is waiting for the target.

If a target becomes available to the current interaction before the timeout elapses, the value associated with the key `return` in the returned list is `ok`, and detailed information on this target is returned in the same format as in the case of low-level targets returned by `SelectDN`. This low-level target can then be used for explicit routing (see `RouteCall` on page 607), possibly after further editing.

Note: The output of this function is usually used as the input for other functions, such as `BlockDN`, `Translate`, `DeliverCall`, or `RouteCall`.

The following errors are possible:

- `-001 Timeout`—wait time is over.
- `0008 Routing done`—not permitted in postrouting.

Note: The BlockDN function (see [page 589](#)) blocks and unblocks a DN for a specified period of time.

SuspendForTreatmentEnd

Parameter: Time: INTEGER or variable

Return value type: STRING (target)

This function pauses the execution of the strategy until the last treatment begun with one of the StartTreatment functions is over or until the end of the interval specified in seconds by the Time parameter.

If no treatment is applied to the interaction, the function immediately returns return:ok.

If the interaction undergoes a treatment that ends before the specified time is over, the function returns return:ok as soon as the treatment ends.

If the interaction undergoes a treatment that continues during the specified interval, the function returns return:timeout when the specified time is over.

The following error messages are possible:

- -001 Timeout—wait time is over.
- 0008 Routing done—not permitted in postrouting.
- 0013 Remote error—T-Server reports an error.

TargetSelectionTuning

Parameters: UseCostFactor: INTEGER (true or false) or variable

Return value type: VOID

Use this function to activate a configured cost-based routing solution. For more information on CBR, see the chapter on cost-based routing in the *Universal Routing 7.6 Cost-Based Routing Configuration Guide*.

Translate

Parameter: Destination: STRING (target) or variable (representing a string for the destination)

Return value type: STRING (target)

This function adds key-value pairs, which correspond to the results of number translation, to the target that is supplied as a parameter. All the key-value pairs with keys other than return are present in the returned result. In the output, the value associated with return can be ok or error. If the return value is ok, the added key-value pairs have the following keys:

- `label`—the value of this key is the translated DN; this value is obtained from the instructions in the DN Source field of the switch-to-switch access code.
- `location`—the value of this key is the translated switch; this value is obtained from the instructions in the Location Source field of the switch-to-switch access code.
- `dnis`—the value of this key is the translated DNIS; this value is obtained from the instructions in the DNIS Source field of the switch-to-switch access code.
- `route_type`—the value of this key is the translated route type represented as an integer; this value is taken from the Route Type field of the switch-to-switch access code.
- `extension`—the value of this key is the translated Extensions list of the interaction; this value is obtained from the instructions in the Extensions Source field of the switch-to-switch access code.
- `reason`—the value of this key is the translated Reasons list of the interaction; this value is obtained from the instructions in the Reasons Source field of the switch-to-switch access code.

If the key return has a value of `error`, the following two error messages are possible:

- `0008 Routing done`—not permitted in postrouting; this message will be returned if the interaction has already been routed successfully by the function `RouteCall` or by a `Routing` object, or if the interaction has been force-routed either by the function `TRoute` or by a combination of the `Force` object with a `Routing` object.
- `0019 Translation failed`—likely due to incorrect configuration; this error message is returned if either the destination switch was not found in the Configuration Database or no access code was configured from the current switch to the destination switch.

If the required configuration options are specified appropriately, URS automatically performs translation. Use this function if you want translation to be done independently of the options set in the URS Application object.

Retired Functions

Retired functions and replacements are presented below.

CountSkillInGroup

This function is no longer used and has been replaced by the function `CountSkillInGroupEx`.

Default

This function has been replaced by the Default Routing object.

GetLastError

This function has been replaced by the Error Segmentation object (see [page 123](#)).

GetSkillInGroup

This function is no longer used and has been replaced by the function GetSkillInGroupEx.

Goto

This function is no longer necessary because the output of an object can be linked to any other object.

Increment

This function is replaced with IncrementPriority.

InvokeNirvanaMethod

This function has been retired.

OnError

This function has been retired. Use the Error Segmentation object (see [page 123](#)).

SetThreshold

This function has been replaced by “SetThresholdEx” on [page 577](#).

SetTreatmentMode

This function has been retired in the 6.x releases. When running 5.1 strategies in URS, use the URS option `compat_treatments` to specify whether treatments will repeat. See “`compat_treatments`” on [page 636](#) for more information.

URS 6.5 still supports the SetTreatmentMode function if it is used in a 5.1.x strategy. In 5.1.x releases, the function SetTreatmentMode overwrote the `compat_treatments` option. If the SetTreatmentMode function is used in a 5.1.x strategy, although the SetTreatmentMode function does not overwrite the

`compat_treatments` option, the function does take precedence over the `compat_treatments` option.

Skill

This function has been retired. See Multiskill function on [page 526](#).



Chapter

4

Configuration Options

Universal Routing provides various configuration options for handling interactions and communicating with other servers. This chapter contains the following topics:

- [Setting Options, page 620](#)
- [Summary of Options, page 621](#)
- [List of URS Options, page 622](#)
- [URS Option Descriptions, page 629](#)
- [Web Services Options, page 687](#)
- [Web API \(URS-Behind\) Options, page 692](#)
- [IRD Options, page 695](#)
- [Custom Server Options, page 697](#)
- [Routing-Related Interaction Server Options, page 699](#)
- [Routing-Related Message Server Options, page 701](#)
- [ADDP Options for Universal Routing Server, page 701](#)
- [Retired Options, page 703](#)

See [page 699](#) for information on options that control the interaction of URS and Interaction Server.

See [page 697](#) for information on Custom Server options.

Note: Log section options are described in the *Framework 8.1 Configuration Options Reference Manual* since log section options control Management Layer logging and are common to all servers. Also see “verbose” on [page 684](#) for information on setting URS log options in the `default` section.

Note: `sml` section options are described in the *Framework 8.1 Configuration Options Reference Manual* since `sml` section options control Management Layer logging and are common to all servers.

Setting Options

Most options described in this section are specified on the `Options` tab of the Properties window of the Application object.

Other options must be added in Configuration Manager via the `Properties` dialog box of servers to which URS connects, Tenants that use the URS, and routing points and other DNs.

URS Options

URS options are placed in the following option folders:

- For the URS Application—in the `default` section of the `Options` tab or the `Annex` tab of the URS Properties dialog box. If options are specified in both places, those specified in the `Options` tab take precedence.
- For a T-Server Application to which URS connects—in the `Options` tab or the `Annex` tab of the T-Server Application Properties dialog box, in a section with the same name as the name of the URS Application, or in a section named `__ROUTER__` or `default`.
- For a Stat Server Application to which URS connects—in the `Options` tab of the Stat Server Application Properties dialog box, in a section with the same name as the name of the URS Application, or in a section named `__ROUTER__` or `default`.
- For a Message Server Application to which URS connects—in the `Options` tab of the Message Server Application Properties dialog box, in a section with the same name as the name of the URS Application, or in a section named `__ROUTER__` or `default`.
- For all other object types—in the `Annex` tab of the object Properties dialog box, in a section with the same name as the name of the URS Application, or in a section named `__ROUTER__`.

To configure options in the `Annex` tab, make sure that the `Annex` tab is displayed. To do this:

1. In Configuration Manager, select `Options` from the `View` menu.
2. Select the `Show Annex tab` check box.

Summary of Options

[Table 136](#) lists the options that you can configure in the Application objects for URS, IRD, routing points, Tenants, T-Server, or other appropriate components (as specified in the full description of the option), a short option description, and the page number where the option information can be found. The table contains brief descriptions of the options, for quick reference.

- For URS options, see [page 622](#).
- For IRD options, see [page 695](#).
- For Custom Server options, see [page 697](#).
- For Message Server options used by URS for monitoring, logging, and load balancing, see [page 699](#).
- For a list of other options that affect communication between URS and the Genesys eServices Interaction Server component, see [page 703](#).
- For the options used to configure the HTTP Bridge functionality, which enables you to contact Web Services from a strategy using the Web Service object, see “Web Services Options” on [page 687](#).
- For log option descriptions, see the Universal Routing Server section of the *Framework Combined Log Events Help*. Also see the `verbose` option on [page 684](#).
- For information on the High Availability licensing feature (`router_ha_option`) set in Licensing Manager, see the *Genesys Licensing Guide*. Also see option “`pickup_calls`” on [page 650](#).
- For a list of retired options, see [page 703](#).

Warning! To avoid problems, you are strongly advised to read the detailed description of the options later in this chapter.

List of URS Options

Table 136 lists URS options in alphabetical order.

Table 136: URS Options

Name	Brief Description	Detailed Description Page
agent_att	Specifies time, in seconds, that URS may use as a value of an agent's average talking time.	629
agent_reservation	Prevents two more URSs from trying to route to the same target.	629
alternative_server	Specifies the name of an alternative Stat Server application for URS to reconnect to.	631
automatic_attach	Controls whether URS automatically attaches pegs to different strategy actions.	632
backup_mode	Provides a choice of setting the backup mode as either “hot” or “standard.”	632
call_kpl_time	One aspect of fixing the problem of calls getting stuck in virtual queues by improving synchronization between URS, T-Server, and various Genesys Reporting components. Stuck calls can happen when connections between these components are broken.	633
call_monitoring	Set this option to true to have URS register on a special DN that enables you to track the number of live calls. Use it when you need URS to supply statistics for non-configured DNs.	634
call_tracking	Controls the way URS tracks interactions.	634
change_tenant	Enables URS to switch the name of the tenant when using the Switch-to-Strategy Routing object.	635
check_call_legs	Controls synchronization of URS with T-Server upon an EventPrimaryChanged.	635
compat_treatments	Determines the mode for busy treatment requests and whether busy treatments should repeat.	636
count_calls	Enables URS to calculate the RStatCallsInQueue and RStatLoadBalance statistics for specified DNs.	637

Table 136: URS Options (Continued)

Name	Brief Description	Detailed Description Page
cpu_emergency_level	Provides the number of seconds which controls activating/deactivating of CPU shortage mode for URS.	637
def_certificate	Certificate to use for secure connection if Web Service object doesn't provide any.	690
def_certificate_key	Certificate key to use for secure connection if Web Service object doesn't provide any.	690
def_trusted_ca	Certificate authority to use for secure connection if Web Service object doesn't provide any.	690
default_db_server	Determines the name of the DB Server that URS considers to be the default.	637
default_destination	Specifies a default destination for routing interactions.	638
default_object	Designates a default routing object.	638
default_stat_server	Designates a default Stat Server.	639
emergency_verbose	Allows reducing URS logging in case a lack of CPU resources is observed.	639
environment	Informs URS about various environment parameters in case there are some actions URS should undertake to guarantee successful functioning in that particular environment.	639
event_arrive	Enables URS to monitor DNs that it usually doesn't monitor (or stop).	640
extrouter_dn	Registers a virtual DN.	641
extrouter_timeout	Specifies the time that URS will wait for a remote access number.	641
function_compatibility	Regulates the compatibility of the behavior of a function.	642
give_treatment	Controls whether URS gives the Ringback treatment to incoming interactions (as long as there is no currently played Busy treatment)	642

Table 136: URS Options (Continued)

Name	Brief Description	Detailed Description Page
hanged_call_time	Specifies the minimum time interval (in seconds) that URS will keep a hung interaction in memory.	643
hide_private_data	Prevents attached data from appearing in the URS log file.	643
hot_backup_priority	Specifies priority of alternative Stat Server to use when the <code>backup_mode</code> option value is set to <code>hot</code> .	643
http_conn_idle_timeout	Time period HTTP Bridge will maintain persistent SOAP connections without activity, in seconds.	691
http_log_file	Log file for HTTP Bridge error and trace messages.	688
http_log_size	HTTP Bridge maximum log file size in bytes.	689
http_port	Use to set communication parameters with the <code>HTTP Bridge</code> component used for Web Services and Workforce Management. Can also be configured to enable Web API (URS-Behind) functionality through REST HTTP API.	688 692
ignore_customer_id	Determine whether URS ignores or considers the <code>customer_id</code> parameter.	644
inv_connid_errors	Specifies a list of T-Server errors that will result in URS aborting a chain of re-routing attempts.	644
lds	Specifies the type of information that URS will exchange.	645
log_buffering	Specifies whether log buffering is performed. If the value of this option is set to <code>true</code> log buffering is on.	689
log_remove_old_files	This option affects only the HTTP log (<code>http_log_file</code>).	689
management_port	Specifies the TCIP/IP for establishing a connection with URS.	645
max_cpu_objects_slice	Specifies the limit (or number of) executed blocks that can be reached, until URS considers a strategy hanged.	646
monitoring_time	Reduces the volume of monitoring data passed from URS to IRD to prevent overloads.	646

Table 136: URS Options (Continued)

Name	Brief Description	Detailed Description Page
null_value	This option is used when a reply from DB Server contains empty (NULL) values.	647
on_primary_changed	This option controls URS behavior when EventPrimaryChanged is received from the T-Server.	647
on_route_error	Instructs URS what action to take if an external error occurs.	648
on_router_activated	Distributes interactions when a backup URS becomes primary URS.	648
pickup_calls	Enables smart registration for routing points.	650
prestrategy	Name of subroutine without parameters that will be executed for every call on this routing point.	651
pulse_time	Specifies the approximate time (in seconds) between consecutive re-checking of target states.	652
reconnect_time	Specifies the time for reconnecting to Configuration Server.	653
reduced	Enables switching off of certain features to increase URS efficiency.	653
reg_attempts	Specifies the number of attempts to register a routing point.	654
reg_delay	Specifies the time that URS delays DN registration.	655
report_reasons	Enables you to add a Reason code to interactions regarding the reason for routing.	655
report_statistics	Supports load balancing reporting, and allows for real-time and historical reporting of calls in transition	658
report_targets	Controls whether URS attaches information to interactions about the targets for which it is waiting.	660
request_timeout	Limits the time that URS waits for a response from DB Server, Custom Server, or Stat Server.	663
reservation_pulling_time	Specifies the time period that URS waits when agent reservation is enabled.	664

Table 136: URS Options (Continued)

Name	Brief Description	Detailed Description Page
route_consult_call	Determines how URS handles consult calls.	664
run_time_mode	Allows URS to adjust its run time behavior.	666
service_timeout	Specifies the error timeout if nothing is received by URS from third-party server (Interaction Server) during the specified time	666
skip_targets	Allows URS to automatically detect a situation when all target agents are logged out and skip waiting time in this case.	667
soap_conn_idle_timeout	Time period HTTP Bridge will maintain persistent SOAP connections without activity, in seconds.	691
soap_port	HTTP port used to access URS's service using SOAP protocol.	693
soap_retry_attempts	Maximum number of attempts HTTP Bridge makes to communicate with a Web Service.	691
soap_retry_timeout	Time period HTTP Bridge waits between retry attempts, in seconds.	691
startup_verbose	Specifies verbose log level output on URS startup.	667
strategy	Instructs URS to run a Genesys-supplied strategy to support multi-Campaign agents.	668
targets_order	Controls default target selection.	668
tattributes	List of attributes that a snapshot is taken for, and accessed by the GetRawAttribute function.	669
Threshold	Specifies the upper limit of interactions for which URS can apply treatments.	669
ThresholdGroup	Imposes limitations on groups of routing points so they share a common threshold.	669
ThresholdSwitch	The switch name where the ThresholdDestination DN is located.	670
transfer_time	Helps URS to spot lost interactions.	671

Table 136: URS Options (Continued)

Name	Brief Description	Detailed Description Page
transfer_to_agent	Instructs to transfer interactions from an IVR directly to a target agent.	671
transit_dn	Instructions URS to register for the DN, but does not enable it to load strategies at the DN.	671
transition_time	Specifies the minimal time that URS waits before routing to the same agent, place, or DN.	672
treatment_delay_time	Prevents URS from applying busy treatments before determining whether a target is available.	672
unix_socket_path	Specifies a socket path to use to connect to Stat Server.	673
unloaded_cdn	Specifies action when an interaction is routed to a DN without a loaded strategy.	673
use_agent_att	Defines conditions for using option <code>agent_att</code> .	674
use_agent_capacity	Specifies whether URS should route interactions based on agent capacity model.	674
use_agentid	Routes interactions to agents based on user ID.	676
use_dn_type	Specifies the default type of DN to be used.	677
use_extrouter	Instructs URS to support various parameters of external routing.	678
use_extrouting_type	Instructs URS which type of external routing to use.	679
use_ivr_info	Instructs URS to start a strategy from the beginning when an interaction is transferred to a routing point controlled by URS.	680
use_parking_threshold	Turns the parking threshold mechanism on and off.	680
use_service_objective	Specifies whether URS should assign baseline Service Objectives to incoming interactions.	681
use_translation	Turns number translation on and off.	682
using	Coordinates access resource selection and prevents congestion when multiple URSs can route calls to the same destination.	683

Table 136: URS Options (Continued)

Name	Brief Description	Detailed Description Page
validate	Instructs URS to validate interactions in a multi-tenant environment.	683
verbose	Specifies level of URS log output.	684
verification_mode	Specifies whether URS will block an agent for specified reservation time.	686
verification_time	Specifies the time between an agent ending an interaction and becoming ready for a new interaction.	686
verification_time_agent	Extends the <code>verification_time</code> option by applying verification time after the agent has ended a call.	687
verification_time_dn	Extends the <code>verification_time</code> option by applying verification time after the agent has ended a call.	687
wfm_polling_interval	Time period in minutes between obtaining agent's schedule data from WFM.	692

The “[URS Option Descriptions](#)” section below lists all URS options. The same option can appear in the properties of different configuration objects.

Location in Configuration Layer By Precedence

Each option description in the pages ahead starts with “Location in Configuration Layer by precedence,” which specifies which option value will be considered by URS as valid when executing an action. The objects for which the option can be specified are listed in order of decreasing priority. For example, assume the priority sequence is specified as DN, T-Server. In this case, URS is guided by the value of the option specified for the DN. If an interaction comes to a DN for which the option has not been specified, and a value for the same option has been specified in a T-Server, that value will be used.

URS Option Descriptions

Also see:

- URS options for Web Services are described on [page 687](#).
- Web API options are described on [page 692](#).
- IRD options are described on [page 695](#).
- Custom Server option descriptions are described on [page 697](#).
- Routing-related Interaction Server options are described on [page 699](#).
- URS options for Message Server are described on [page 701](#).
- ADDP options are described on [page 701](#).
- Retired options are described on [page 703](#).

agent_att

Location in Configuration Layer by precedence: Virtual Queue, URS

Default value: `0`

Valid values: any positive integer

Value changes: take effect immediately

This option specifies the time, in seconds, which URS uses as a value of an agent's average handling time of voice interaction.

If the option is set to `0`, URS uses a hardcoded value of 150 seconds until the agent completes the first call, and after that the calculated average handling time is used.

The option `agent_att`, along with its related option `use_agent_att`, [page 674](#), are used for calculating metrics returned by the function `RvqData`, [page 570](#).

agent_reservation

Location in Configuration Layer by precedence: T-Server, URS

Default value: `false`

Valid values: `true`, `false`, `implicit`

Value changes: take effect immediately

This option enables two or more Universal Routing Servers to work cooperatively and prevents them from trying to route to the same target. To turn agent reservation on, this option must be `true` for URS.

Note: If you have URSSs routing to the same set of agents, agent reservation is mandatory.

In this case, URS will look for the best T-Server to make the agent reservation, which is a T-Server with this option (`agent_reservation` in the section with the

name of the URS) set to true. T-Servers with this option set as false cannot be used for agent reservation. If there is no T-Server with this option set as true, URS will select the T-Server that controls the agent DN.

In T-Server options, URS looks for the section with its own name. If URS finds such a section, it takes its own agent reservation options from this section (if any). The agent reservation options for T-Server, can be found in the `agent-reservation` section. The options are `request-collection-time`, `reservation-time`, and `reject-subsequent-request`.

URS sends a reserve request to the T-Server and sends the interaction only when the request is confirmed. When a T-Server reserves an agent, place, or DN, it prevents any other URS from sending an interaction. The target is reserved for a duration equal to the value of `transition_time`.

Note: If `reservation_pulling_time` is set to 0 (default) URS will not wait for a reservation response in any target Selection object with a wait time set to 0. As a result, the target Selection object will timeout even if a target was found.

- For additional information on `agent_reservation` option, see the *Universal Routing 8.1 Deployment Guide*, “System Availability and Redundancy” chapter. Also see the section on Router Self-Awareness in that guide for information on how this mode can affect agent reservation.
- For information on how T-Server clients set agent reservation priority, see the *Genesys Events and Models Reference Manual*.
- See also: “`reservation_pulling_time`” on [page 664](#) and Table 140 on [page 700](#) for agent reservation options set in Interaction Server.

Important Information

- You have two ways of configuring agent reservation: agent reservation through Inter Server Call Control (ISCC) and centralized agent reservation.
Agent reservation through ISCC: Set the `agent_reservation` option as `implicit`. In this case, URS will include the agent reservation request into the standard routing request, such as `RequestRouteCall` or `RequestGetAccessNumber`.
Centralized agent reservation. To activate, set the URS option `agent_reservation` to `true`. In addition, the URS option `agent_reservation` must be set to `true` on the dedicated T-Server.
- When configuring a multi-tenant environment for requesting agent reservation from one centralized T-Server, select the centralized T-Server from T-Servers configured under same tenant or under Environment tenant. If Agents, Places and Groups are configured under Environment tenant, then selection of the centralized T-Server is limited to T-Servers configured under the Environment tenant only.

- When the agent reservation feature is used in an LDS/multi-URS environment, Genesys recommends that all URSs come from the same family (all 7.5 or all 7.6, for example), and that you configure them all with the same set of options.
- If Router Self-Awareness is on (as described in the *Universal Routing 8.1 Deployment Guide*), then every URS will block an agent for routing as soon it receive notification that this agent was selected by some other URS. The assumption here is that other URS's notification can arrive much sooner than the agent will be reported as busy by Stat Server. This can save URS from the necessity of doing a reserving request that will unconditionally fail.
- URS will make a reservation request to IVR targets if agent reservation is configured (enabled).

T-Server Selection

URS counts T-Server's score for selecting the most suitable T-Server for agent reservation by adding up the following:

- TServer application objects having the option `agent_reservation` set to `false`, or belonging to the wrong tenant, except Environment tenant, are not considered.
- If the TServer application object has the `agent_reservation` option set to `true`, it gets 4 points.
- If the T-Server's switch is the same switch as the DN number to be reserved, it gets 3 points.
- If the T-Server's switch isn't the same switch as the DN number to be reserved, but the agent has some other DN from this T-Server's switch, it gets 2 points.

alternative_server

Location in Configuration Layer by precedence: Stat Server Application object URS is connected to.

Default value: `none`

Valid values: name of Stat Server application

Value changes: take effect immediately

Related feature: the possibility to connect to an alternative Stat Server as source of statistic information. For a description of configuration, see “Reconnection to an Alternative Stat Server” in the *Universal Routing 8.1 Deployment Guide*.

Related options: `backup_mode` on [page 632](#), `hot_backup_priority` on [page 643](#).

Specifies the name of the alternative Stat Server application for URS to use as an alternative source of statistic information for the period of time when the main source of statistic information is unavailable. The alternative server plays

the role of a backup server for the application where the `alternative_server` option is specified.

Configure this option on the primary Stat Server Application object which is defined on the Connections tab of URS Application object.

In a deployment where the primary Stat Server application has backup configured on the Server Info tab, configure this option on the backup Stat Server Application object.

automatic_attach

Location in Configuration Layer by precedence: URS

Default value: `true`

Valid values: `true`, `false`

Value changes: take effect immediately

When set to `true`, URS automatically attaches pegs (see [page 711](#)) corresponding to different actions in the strategy. When set to `false`, URS makes no automatic attachments.

backup_mode

Location in Configuration Layer by precedence: Stat Server application URS is connected to.

Default value: `standard`

Valid values: `standard`, `hot`

Value changes: take effect on next reconnect to the server where option is specified.

Related feature: the possibility to connect to an alternative Stat Server as source of statistic information. For a description of configuration, see see “Reconnection to an Alternative Stat Server” in the *Universal Routing 8.1 Deployment Guide*.

Related options: `alternative_server` on [page 631](#), `hot_backup_priority` on [page 643](#).

This option specifies the HA redundancy type between the pool of Stat Server applications which is defined as a set of primary and backup pair of the main Stat Server, together with primary and backup pairs of alternative Stat Server applications.

When set to `standard`, URS connects to one server only and reconnects to another server upon disconnect.

When set to `hot`, URS connects to all Stat Server applications in the pool simultaneously.

call_kpl_time

Location in Configuration Layer by precedence: URS

Default value (in seconds): 60

Valid values: No less than 10 seconds with the exception of 0 (zero means no “keep alive” notifications)

Value changes: take effect immediately

This option addresses the potential problem of calls getting stuck in virtual queues by improving synchronization between URS, T-Server, and various Genesys Reporting components (ICON, Stat Server, and so on). Stuck calls can happen when connections between these components are broken. This potential problem is addressed by URS and Stat Server as follows:

URS:

- In addition to the basic virtual queue events (such as EventQueued and EventDiverted), URS now supports EventStillInQueue.
- URS extends all virtual queue event messages (including EventStillInQueue) with a time parameter, MaxTimeForNextEvent. This parameter delimits the time in seconds that the next virtual queue event for a call will be distributed. T-Server distributes events no later than this time. URS gets MaxTimeForNextEvent via option call_kpl_time, which by default is 60 seconds. At the URS level, this option is dynamic. Any new value is applied to subsequent new calls and not to old calls.
- In the time interval between sending EventQueued and EventDiverted for a call, URS sends EventStillInQueue for every MaxTimeForNextEvent number of seconds. The MaxTimeForNextEvent event parameter tells listening applications (Stat Server) when to expect the next virtual queue event.

Stat Server:

- Stat Server removes a call from a virtual queue (considers it as a stuck call) when it doesn't receive EventStillInQueue for the call and virtual queue for a time period indicated by the MaxTimeForNextEvent parameter starting from the last received virtual queue message for a call.

This combined URS/Stat Server behavior allows:

- The restoration of calls in virtual queue after T-Server/Stat Server and URS/T-Server reconnection within a configurable time interval.
- The cleanup of stacked calls in the case where URS stops or shuts down unexpectedly.

Note: If there are calls in queue, it is possible to dynamically switch off the events (set call_kpl_time option value to 0), but it is not possible to switch on the events for existing calls by changing the call_kpl_time value again.

call_monitoring

Location in Configuration Layer by precedence: T-Server, URS

Default value: `false`

Valid values: `true, false`

Value changes: On connection to T-Server

When set to `true`, registers a special DN, `switch::`, which enables you to track the number of live calls. Used if you need the URS-calculated statistic `RStatCallsInQueue` for non-configured DNs.

call_tracking

Location in Configuration Layer by precedence: routing point, T-Server, tenant, URS

Default value: `true`

Valid values: `true, false`

Value changes: take effect immediately

Controls the way URS tracks interactions transferred by URS from the routing point to an IVR and vice versa. When the option is set to `true`, URS waits 60 seconds after transferring an interaction for a T-Server event confirming that the interaction has reached its destination. If no such event is received from the destination point at the end of the waiting time, URS deletes the interaction. No such limit is set if the option has a value of `false`.

The `call_tracking` option is used to delete interactions that URS is passing between an IVR and a route point that have not changed from a particular state for more than 60 seconds. It is intended to clean up inactive interactions. The mechanism is activated once every 60 seconds and checks every interaction currently controlled by URS.

Note: The value of `call_tracking` should be set to `false` if IVRs are used for a DN for which URS is not registered.

The interaction is deleted (or, in the case of an inactive database query, default routed) if, at the time of the check:

- It is controlled by URS, but more than 60 seconds have passed and URS has not started a strategy for the interaction.
- A routing instruction is issued for the interaction as a result of the `RouteCall` function or a `Routing` object, but more than 60 seconds have passed and `EventRouteUsed` has not arrived for the interaction.
- A database query has produced no results after more than 60 seconds. In this case, the interaction is sent to the default destination instead of being deleted.

- URS inquired for a remote access number in order to deliver the interaction to a DN on a remote switch, but more than 60 seconds have passed and the external router has not returned an access number.
- URS ordered the interaction to be delivered to another DN, but more than 60 seconds have passed and no event confirming the interaction's departure from the current DN has been received.
- URS ordered the interaction to be delivered to another DN, but more than 60 seconds have passed and no event confirming the interaction's arrival has been received.

Note: URS does not detect call abandonment during a mute call transfer back to the Routing Point. URS now considers EventReleased events with a CallState attribute equal to 0, an indication of the opposite party having left the call. Previously, after a call transfer was initiated and T-Server confirmed that the transfer was started, (and before the transfer was completed) URS considered the EventReleased event a part of the transfer. Therefore, when an EventReleased event was received in the middle of the transfer, URS activated stuck call detection and these calls were deleted.

change_tenant

Location in Configuration Layer by precedence: URS

Default value: `false`

Valid values: `false`, `true`

Value changes: take effect immediately

The Switch to Strategy Routing object (see [page 347](#)) enables you to select the Tenant name associated with the switched-to strategy. In order to use this functionality, you must have access to the switched-to tenant's Configuration Manager Environment, all tenants must be on the URS tenant list (Tenants tab), and the URS `change_tenant` option must be set to `true`.

check_call_legs

Location in Configuration Layer by precedence: routing point, T-Server, URS

Default value: `default`

Valid values: `default`, `true`, `false`

Value changes: take effect immediately

This configuration option controls synchronization of URS with T-Server when `EventPrimaryChanged` is received. The option is dynamic, and can be defined on the routing point, T-Server, or URS levels.

A value of `true` indicates that URS will perform synchronization (check if calls are still present on the routing point) upon receiving `EventPrimaryChanged`.

A value of `false` indicates that URS will not perform synchronization upon receiving `EventPrimaryChanged`.

A value of `default` indicates that URS will perform synchronization upon receiving `EventPrimaryChanged`, but only for calls not having `AttributeUserEvent` in the `EventRouteRequest` event.

compat_treatments

Location in Configuration Layer by precedence: T-Server, URS

Default value: `true`

Valid values: `true`, `false`, `dynamic`

Value changes: take effect immediately

Formerly, this option had two actions:

1. The first action determined the way URS requests T-Server to begin treatments (through `RequestApplyTreatment` or requests like `RequestGiveRingBackTreatment`, etc) in cases where a strategy does not provide this information. Starting with release 6.x, this action has limited usage as strategies always explicitly provide information on how to play treatments. The only exception is playing the ringback treatment that is enabled with the URS `give_treatment` option.
2. The second action of the option determined whether or not busy treatments should repeat (in other words, whether or not URS should start playing the sequence of busy treatments in the `Busy` tab of a `Routing` object from the beginning when it finishes the sequence if the interaction is still waiting).

true/false/dynamic Actions

- When set to `true` (or in release 5.x/6.x, strategies where `SetTreatmentMode` is `Meridian`) URS was forced to send treatment requests by means of the T-Library functions `TGiveMusicTreatment`, `TGiveSilenceTreatment`, and `TGiveRingBackTreatment` for treatments other than IVR, and by means of a special set of URS functions for IVR treatments. This is only applicable in cases where a strategy does not provide the information on which TLib API to use. A setting of `true` also causes busy treatments to repeat themselves.
- When set to `false` (or in case of 5.x/6.x strategies if `SetTreatmentMode` is `Network`), URS sent treatment requests by means of the `TApplyTreatment` function. This is only applicable in cases where a strategy does not provide the information on which TLib API to use. A setting of `false` also causes busy treatments to NOT repeat themselves.
- When set to `dynamic`, URS queries T-Server on the supported treatment API and the sent treatment requests accordingly. This is only applicable in cases where a strategy does not provide the information on which TLib API to use. A setting of `dynamic` also causes busy treatments to repeat themselves.

The `compat_treatments` option can be in either the T-Server or URS Application. If it is set in both Applications, the value in the T-Server application is used.

count_calls

Location in Configuration Layer by precedence: Any destination DN where URS registered (either by default or because of some other option), URS

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect immediately

If set to `true`, instructs URS to track number of calls on one or more DNs. Can be applied only to DNs which URS is registered on. Used to enable URS to calculate the `RStatCallsInQueue` and `RStatLoadBalance` statistics for the specified DNs.

cpu_emergency_level

Location in Configuration Layer by precedence: URS.

Default value: `4`

Valid values: any positive integer (any value higher than `10` is counted as `10`)

Value changes: take effect on restart

Option specifies the time interval, in seconds, which is used to determine if URS is experiencing a shortage of CPU resources. If a shortage is detected, URS switches to CPU saving mode.

This option is activated when the `reduce` option is set to `2048`.

See the *Universal Routing 8.1 Deployment Guide* for more information.

default_db_server

Location in Configuration Layer by precedence: routing point, T-Server, tenant, URS

Default value: `none`

Valid value: the name of any available DB Server DAP

Value changes: take effect immediately

When the user fails to specify a DB Server in the `Database Wizard Properties` dialog box (see Figure 51 on [page 108](#)), a special parameter `__DEFAULT__` is automatically inserted in the Database object or special function in the strategy.

The parameter `__DEFAULT__` indicates that the DB Server specified in the URS configuration option `default_db_server` should be used for the interaction.

Note: In the case of having multiple DAPS in URS's Connections list and wanting to control which DAP will be used by `__DEFAULT__`, you must set `default_db_server` to the name of the DAP that you want to use (it must be one that is in URS's Connections list). Even though the option name is `default_db_server`, you must specify the name of the DAP and not the DAP's DB Server.

default_destination

Note: This option can be overridden by the SetCallOption function. See [page 475](#) for more information.

Location in Configuration Layer by precedence: routing point, T-Server, tenant, URS

Default value: none; a value must be supplied to ensure predictable behavior

Valid values: configured DNs

Value changes: take effect immediately

Defines a default destination for routing interactions if the routing strategy fails to route the interaction.

Notes: Default DNs can also be set at the routing point to control the default for each strategy. This is done by creating a shortcut to the corresponding routing point in the `Default DNs` tab of the routing point. In this case, more than one default destination for the same routing point can be specified—URS will pick one at random when it has to send an interaction to the default destination.

When executing a strategy where force (TRoute function) or default routing is applied to an interaction, URS removes the call from all virtual queues and only then executes the force or default routing.

Also see “Default” on [page 316](#) and “Default Routed Calls” on [page 730](#).

default_object

Note: This option can be overridden by the SetCallOption function. See [page 475](#) for more information.

Location in Configuration Layer by precedence: routing point, T-Server, tenant, URS

Default value: `agent`

Valid values: `agent`, `agent_place`, `agent_group`, `queue`, `place_group`, `route_point`, `destination_label`

Value changes: take effect immediately

Designates a default routing object to be used when a target in a strategy omits the type of Routing object (that is, the target has a format of ID or ID@StatServerName, rather than ID@StatServerName.type). It is recommended that a default destination be configured using the default object.

default_stat_server

Location in Configuration Layer by precedence: routing point, T-Server, tenant, URS

Default value: the first available Stat Server

Valid value: the name of any available Stat Server

Value changes: take effect immediately

Designates which Stat Server to use as the default location when a target in a strategy omits the location (that is, the target has a format of ID or ID.type rather than ID@StatServerName.type).

emergency_verbose

Location in Configuration Layer by precedence: URS

Default value: 5

Valid values: from 0 to 5

Value changes: take effect immediately

Allows reducing URS logging in case a lack of CPU resources is observed. If the value of this option is numerically less than the value of the option verbose, then upon detecting a shortage of CPU resources, URS will temporarily switch its logging to the emergency_verbose level, and restore it back to the verbose level when the CPU shortage is over.

environment

Location in Configuration Layer by precedence: URS

Default value: none

Valid values: Comma-separated list of environment parameters that URS needs to be aware of. The only valid parameter is outbound

Value changes: take effect on restart

The purpose of this option is to inform URS about various environment parameters in case there are some actions URS should undertake to guarantee successful functioning in that particular environment.

Use Case for outbound Value

URS and Outbound Contact Server (OCS) can cooperate for the purpose of sharing multi-skilled Agents and/or Places. A special statistic

`CurrentAgentAssignment`, provides information about the activity that the Agent/Place is currently busy with (assigned to). Activities can be related to inbound interactions or Campaign Group activities. Use the environment option to ensure that an Agent/Place busy with one of these activities does not get any calls associated with any other activities.

By default, URS opens the `CurrentAgentAssignment` statistic for Agent/Places from a Campaign Group specified as target in a strategy (Routing Selection object or Route Interaction object). Since URS does not know the current activity for all other Agent/Places, it is possible (if those multi-skilled Agents participate in multiple activities) for URS to send calls associated with other activities.

Setting the environment option to `outbound` notifies URS that *all* Agents/Places in its configuration environment can potentially participate in some Outbound Campaign (not just those that are members of some Campaign Group used in a strategy). In this case, URS opens the `CurrentAgentAssignment` statistic for *all* Agents or Places. As a result, *all* Agents/Places become protected from calls related to different activities no matter how they are addressed in strategies.

For more information, see “Campaign Group Targets” on [page 362](#).

event_arrive

Location in Configuration Layer by precedence: routing point, T-Server, URS
Default value: none

Valid values: `none`, `routerequest`, `ringing`, `established`, `treatmentrequired`
Value changes: take effect immediately

This option, which can be used for load balancing in an Enterprises Routing solution as described in the *Universal Routing 8.1 Deployment Guide*, enables URS to monitor DNs that it doesn’t usually monitor or to stop monitoring DNs that it usually monitors. If URS needs to control a DN of a different type or to start processing (run the strategy) an interaction upon receiving `EventTreatmentRequired` from an Electronic Audio (EA) Port, the `event_arrive` option must be used for the corresponding DN. This option must be placed in the Annex tab of the DN, in a folder bearing the name of the URS Application object.

In some situations, an EA Port reports its readiness to play treatments with `EventTreatmentRequired`. If this occurs, URS must wait for this event before requesting that the treatment be played. By default, URS requests treatments on `EventEstablished`. If `event_arrive` was not configured, an error would occur since EA Port would not send the `EventEstablished` message.

If the option `event_arrive` is specified with a value of `none`, URS will not control the corresponding DN. One reason you would specify `none` for this option is if another URS controls the DN. You can also use `none` if you want URS to send requests to some T-Server, but not to react to any events received from the T-Server.

If the value of the option is `routerequest`, `established`, `ringing`, or `treatmentrequired`, URS will control the DN and will start processing an interaction upon receiving the corresponding event from the DN. The same kind of processing (start or continue the strategy) occurs for each value.

Regardless of whether `routerequest`, `established`, `ringing`, or `treatmentrequired` is specified, when a strategy is loaded on a DN (CDN or EA Port) and URS receives the specified message based on the value set for the option, URS starts the loaded strategy for the interaction.

Important Information

- For DN types `RoutePoint`, `VirtualRoutePoint`, `ServiceNumber`, or `RoutingQueue`, specify only `routerequest` or `none`. Do not use `routerequest` for any other type of DN. The ringing, established, and treatment required events are not possible for these types of DNs. Only `routerequest` is expected for these types of DNs.
- For DNs with the `event_arrive` option configured, IRD only supports loading strategies to the following DN types: `RoutingPoint`, `VirtualRoutingPoint`, `ServiceNumber`, `EAPort` (voice treatment), and `RoutingQueue`.

Note: For additional information on this option, see the *Universal Routing 8.1 Deployment Guide*, “System Availability and Redundancy” chapter. Also see function “SendEvent” on [page 532](#).

extrouter_dn

Location in Configuration Layer by precedence: T-Server, URS

Default value: `exr`

Valid value: any value is valid

Value changes: take effect immediately

After connection to any T-Server, this option will register a virtual DN. This DN is used to communicate with T-Server out of context of any interaction, for example, agent reservation or external routing. By default, a DN with a value of `exr` is used. This `exr` value can be overridden with this option.

Note: External routing for T-Servers is now called Inter Server Call Control (ISCC).

extrouter_timeout

Location in Configuration Layer by precedence: T-Server (T-Server that is being requested by URS for a Switch Access Code), URS

Default value: `0`

Valid value: any non-negative integer

Value changes: take effect immediately

Specifies the time, in milliseconds, that URS will wait for an access number (`RequestGetAccessNumber`) after requesting it from a local or remote T-Server according to option `use_extrouter` (see [page 678](#)). A value of `0` indicates that no limit is set. If the value of this option is positive, and the timeout elapses without an access number arriving from the external router, URS will issue a routing instruction to the existing DN of the target. In general, this should result in the interaction arriving at the right destination, but without its attached data.

Note: This timeout also applies when the remote access number is requested by the URS function `GetRemoteAccessCode` (see [page 594](#)).

function_compatibility

Location in Configuration Layer by precedence: URS only

Default value: `6.1`

Valid values: `6.1, 6.5, 8.0`

Value changes: take effect on restart

This option regulates the compatibility of the behavior of a function (by customer request, for example). The new behavior of the function is only active if this option has a value that is not less than the URS version in which the function was originally introduced.

In `6.5`, this option only affects the behavior of the `SData` function (see [page 572](#)). If you leave the default value of `6.1` and the connection between Stat Server and URS is broken, `SData` returns `0` and the interaction flows to the green port. If you specify a value of `6.5` and the connection between Stat Server and URS is broken, `SData` also returns `0` but the interaction flows to the red port of the function object rather than the green port.

Otherwise, `SData` functions as it did in previous releases when the connection between Stat Server and URS is broke; interactions flow to the green port.

A value of `8.0` changes the behavior of the `ListGetDataCfg` and the `UData` functions. A value of `8.0` disables the interpretation of an empty key in this function as a command to return all the properties of the `List` element (see “`ListGetDataCfg`” on [page 482](#) and “`UData`” on [page 465](#)). To obtain all the properties when the option is set to `8.0`, the key “`*`” (star) needs to be used. If an empty key is used in this case, the function will return an empty value.

give_treatment

Location in Configuration Layer by precedence: T-Server, URS

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect immediately

When set to `true`, as long as there is no currently played Busy treatment, URS gives each incoming interaction the Ringback treatment and suppresses default routing by the switch. When set to `false`, the switch routes an interaction if URS fails to handle it.

hanged_call_time

Location in Configuration Layer by precedence: URS

Default value: `60`

Valid values: Any positive number or `0`

Value changes: Takes effect immediately

Specifies the minimum timeout interval, in seconds, that URS keeps hanged interactions in memory. This option is dynamic and it does not affect the interval length.

Every 60 seconds, URS checks for hanged interactions against the value of this option. In other words, hanged interactions will be deleted no sooner than the value of `hanged_call_time` and no later than the value of `hanged_call_time` plus 60 seconds.

hot_backup_priority

Location in Configuration Layer by precedence: Stat Server application URS is connected to.

Default value: `1`

Valid values: any positive number

Value changes: Takes effect on next reconnect to the server where option is specified.

Related feature: the possibility to connect to an alternative Stat Server as a source of statistic information. For a description of configuration, see see “Reconnection to an Alternative Stat Server” in the *Universal Routing 8.1 Deployment Guide*.

Related options: `alternative_server` on [page 631](#), and `backup_mode` on [page 632](#).

Specifies priority of Stat Server application for URS to use as alternative source of statistic information. This option should be configured when `backup_mode` option is set to value `hot`. The server with lowest value has priority.

hide_private_data

Location in Configuration Layer by precedence: URS

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect upon restart

Note: All options affecting the `http_bridge` application (including `hide_private_data`) take effect upon restarting `http_bridge` (or URS).

Prevents URS from showing attached data, such as sensitive business data, in the log of T-Library events, configuration data, results of access to database, and values of variables in strategies. In addition, when `hide_private_data` is set to `true`:

- URS will not print in its logs parameters for Web Service requests and also any result that a Web Service returns.
- `http_bridge` will not print in its log parameters and results of Web Service requests. For more on `http_bridge`, see the *Universal Routing 8.1 Deployment Guide*.
- `http_bridge` will not generate an `xml-soap` log.

For more on Web Services, see Appendix B, “IRD Web Service Object,” on [page 771](#).

ignore_customer_id

Location in Configuration Layer by precedence: T-Server, URS

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect immediately

This option determines whether the `customer_id` parameter (if present) in the `AttributeCustomerID` field of the `EventRouteRequest` message is ignored or considered when routing an interaction. When set to `true`, URS ignores the `customer_id` parameter and assumes that the tenant which has responsibility for the routing point is also the tenant to which the interaction belongs. (This is the case in a single tenant environment.) When set to `false`, URS uses the `customer_id` parameter to identify which tenant that the interaction belongs to.

inv_connid_errors

Location in Configuration Layer by precedence: URS

Default value: none (empty)

Valid values: comma-separated list of T-Server errors

Value changes: Take effect immediately

This option allows you to explicitly specify a list of T-Server error codes that will result in URS aborting a chain of re-routing attempts. URS’s reaction to the comma-separated list of error codes is the same as its reaction to the hardcoded error, `TERR_INV_CONNID`: When option `on_route_error` is set to

`try_other`, URS makes one additional attempt to route the call. If the attempt fails, URS then deletes the call.

Note: This option only works for interactions routed to destinations specified in the Routing Selection object. The option does not work for interactions routed to destinations which use the `SelectDN` and `RouteCall` functions.

lds

Location in Configuration Layer by precedence: Message Server

Default value: none

Valid values: `ar` ([access resources](#)), `c iq` ([calls in queue](#)), `b lk` ([agents blocking](#))

Value changes: take effect on restart

Note: Specify this option in the `Annex` tab of the Message Server Application object in the folder with name `__ROUTER__` or with the name of the URS Application object.

If using a Message Server dedicated to handling communications between a group of URSs (in this case, option `using` has the value of `lds`), then use option `lds` to indicate the specific type of communication:

- Use `ar` if this Message Server coordinates work with access resources (DNs). For example, during IVR Server Load Balancing when operating in the mode as described in the *Universal Routing 8.1 Deployment Guide*.
- Use `c iq` if this Message Server coordinates information about available targets and calls in transition (see “Router Self-Awareness” on [page 726](#)).
- Use `b lk` if this Message Server coordinates between URSs the information regarding agents blocking – agents are blocked every time they are selected by a URS for routing a call (see “Router Self-Awareness” on [page 726](#)).
- Any combination of the values can also be used (any number of them, in any order) to make URS use the Message Server for multiple purposes.

Also see option “`using`” on [page 683](#).

management_port

Note: URS no longer provides route point related data through option `management_port`. It prevents CPU overload usage during the reading of configuration data (for example, when a new tenant is added to the URS Application).

Location in Configuration Layer by precedence: URS

Default value: none

Valid value: any valid TCP/IP port used by URS

Value changes: take effect on restart

Specifies the TCP/IP port where an SNMP option client can establish connections with URS. This must be different from the port specified in the **Server Info Properties** dialog box. An arbitrary integer different from the server port can be used.

max_cpu_objects_slice

Location in Configuration Layer by precedence: URS

Default value: 10 (in 10,000 increments)

Valid values: Any positive value

Value changes: take effect immediately (dynamic)

This option is used to configure the limit (or number of) executed blocks that can be reached, before URS considers a strategy hanged. The number of executed blocks allowed is the value of this option multiplied by 10,000. In 8.1.1 and later releases, the router counts the executed blocks throughout the subroutines boundaries `max-submitted-interactions`.

See “Routing-Related Interaction Server Options” on page 699.

max-submission rate

See “Routing-Related Interaction Server Options” on page 699.

monitoring_time

Location in Configuration Layer by precedence: URS

Default value: 5 (if set to 0, no monitoring information is sent)

Valid value: any non-negative integer

Value changes: take effect immediately

This option allows you to specify the time interval (in seconds) that URS uses for sending monitoring data to IRD. Previous to 7.1, this option was set internally to 5 seconds and could not be changed.

Use this option only if you need to reduce the amount of monitoring data passed from URS to IRD. In large environments, this can reduce the chance of URS losing its Message Server connection when large amounts of monitoring data need to be passed to IRD.

Background: If there is a slow network connection between IRD and Message Server or slow IRD host performance, messages may come in to IRD faster than they can be read and accumulate in Message Server. As Message Server waits for its output queue to IRD to be cleared, it slows down receiving messages from URS. At some point, URS becomes unable to send messages because Message Server does not read them. In this

scenario, a disconnect can occur between URS and Message Server. Use of the `monitoring_time` option can prevent this scenario.

Note: This option affects only monitoring data. Information about the state of URS's connections is still sent every 5 seconds.

null_value

Location in Configuration Layer by precedence: DBServer, Database Access Point

Default value: " " (space)

Valid values: any string

Value changes: take effect immediately

This option is used when a reply from DB Server contains empty (NULL) values. Because URS must return a value as a result of DB Server request, URS will use the value of this option for NULL values.

Note: This option can be overridden with the SetCallOption function (see [page 475](#)).

on_primary_changed

Location in Configuration Layer by precedence:

Default value: ignore

Valid values: ignore, reconnect, error

Value changes: take effect upon URS restart

This option controls URS behavior when EventPrimaryChanged is received from the T-Server.

The values are defined as follows:

- A value of ignore indicates that URS will take no action regarding currently active T-Server requests, and will continue to wait for an answer from the new primary T-Server. This is the default value and describes the behavior of URS in previous releases.
- A value of reconnect indicates that URS will attempt to reconnect to the T-Server (effectively switching Hot T-Server Standby to Warm Standby). URS restarts all running strategies on this T-Server from the beginning. This is the most reliable way to recover from a PrimaryChanged event.
- A value of error indicates that URS will imitate an EventError for every pending request. Strategies for all affected calls will behave exactly as if URS received an actual EventError for every pending request. This is a less disruptive way to recover from a PrimaryChanged event than using the reconnect value, but it is limited only to ApplyTreatment and ReserveAgent T-Server requests.

Note: Currently URS does not unblock the target if it was previously blocked as part of an issuing request (for example, ReservAgent), and the error or reconnect occurred as a result of using the `on_primary_changed` option.

on_route_error

Note: The OnRoute Error function (see [page 527](#)) can overwrite option `on_route_error` for the current interaction. The ExtrouterError function (see [page 518](#)) can change URS's default reaction in the case of a failure to get a remote access number.

Location in Configuration Layer by precedence: T-Server, URS

Default value: `delete`

Valid values: `delete`, `ignore`, `default`, `reroute`, `try_other`, `strategy_ok`, `strategy_error`

Value changes: take effect immediately

Instructs URS which action to take if an external error, such as `EventError` from T-Server, occurs after URS issues an instruction for definitively routing the interaction as a result of a Routing object.

This option only works for interactions routed to destinations specified in the Routing Selection object (see [page 326](#)). The option does not work for interactions routed to destinations which use the `SelectDN` and `RouteCall` functions.

The values are defined as follows:

- `Delete` means that URS will delete the interaction and stop the strategy. URS will assume that the interaction was not routed successfully. At this point, nothing can be done to reroute the interaction.
- `Ignore` means that URS will disregard the error and continue waiting for the `EventRouteUsed` event that confirms successful routing.
- `Default` means that URS will attempt to send the interaction to a default destination.
- `Reroute` means that URS will try to route the interaction again on the basis of the same Routing object. URS assumes that the interaction is still at the routing point and will immediately reissue `TRouteCall`.
- `try_other` means that URS will resume waiting for a ready target and try to select another available target. Unlike the `reroute` value in which URS immediately attempts to reroute the call, for `try_other` URS waits for another target if none are available before routing the interaction.

Note: Be sure that the option `transition_time` (see [page 672](#)) has a value of 3 or higher to enable URS to handle the `try_other` setting correctly.

- `Strategy_ok` means that URS will abort current waiting for a ready target and continue with the strategy. In this case, the error from T-Server will have the same effect as absence of a ready target—URS simply resumes the strategy to the error-handling object.
- `Strategy_error` means that URS will abort current waiting for a ready target and go to error handling in accordance with the Error object.

For strategies created in IRD 6.x and 7.x, `strategy_ok` and `strategy_error` do the same thing. For strategies created in Strategy Builder, the strategy continues to the `OnError` block or is default-routed if no block exists.

Since `try_other`, `strategy_ok`, and `strategy_error` can use error-handling objects, you can use this ability to print the error or loop back to a Routing object.

Note: The option `on_route_error` is activated under circumstances very different from the Error object. The option accounts for errors during definitive routing that occur outside of URS. The Error object accounts for errors occurring inside URS during the execution of the strategy. To make the Error object work with the Routing objects (except Default), set the value of this option to `try_other`, `strategy_ok`, or `strategy_error`.

on_router_activated

Location in Configuration Layer by precedence: URS

Default value: `default`

Valid values: `default`, `route`, `ignore`

Value changes: changes take effect immediately

When a backup URS becomes the primary URS, interactions received when the URS was the backup server that were not handled by the old primary URS must be distributed. This option provides a choice of:

- Sending the interactions to the default destination (option value = `default`).
- Replaying the strategy from the beginning (option value = `route`).
- Ignoring (option value = `ignore`), which instructs URS to do nothing with the old interactions.

The last value (`ignore`) is used when handling interactions is the exception rather than the rule (for example, in ring-no-answer scenarios where some router is dedicated to agents DN tracking).

The option correlates with URS's Redundancy type as specified in Configuration Server in following ways:

- If Redundancy type is Hot Standby, then URS does not use this option and replays the strategies.
- If Redundancy type is Warm Standby, then URS uses this option for choosing between `default` and `ignore`. Stated another way, when URS Warm Standby mode is configured, the valid values for option `on_router_activated` are `default` and `ignore`.
- If Redundancy type is not specified or is Cold Standby, then URS entirely follows this option.

Note: For additional information on this option, see the *Universal Routing 8.1 Deployment Guide*, “System Availability and Redundancy” chapter.

pickup_calls

Location in Configuration Layer by precedence: Annex properties of DN controlled by URS

Default value: `false`

Valid values: `false`, `true`, `reverse`

Value changes: changes take effect immediately

This option enables smart registration for routing points and depends on T-Server's ability to provide information on all interactions pending on a routing point even before routing points are registered by URS at startup.

- When `pickup_calls` is set to `true`, URS requests from T-Server information on all interactions reported in `EventRegistered` in the same order as they were reported by T-Server. Upon receiving information on an interaction, URS (if set to `Primary`) will route the interaction.

When `pickup_calls` is set to `reverse`, URS requests from T-Server information on all interactions reported in `EventRegistered` in an order opposite to order they were reported by T-Server.

When reporting the order of calls, some T-Servers put the oldest calls at the end of the list. Setting `pickup_calls` to `reverse` compensates for such behavior and causes the oldest calls to be handled first.

This option has two purposes depending on the whether or not a backup URS is configured:

- For a URS with no backup—when URS restarts after an unexpected shutdown, this option enables URS to pick up existing interactions at the routing points before URS registers the routing points and attempts to route the interactions.

- For environments with a backup URS—during the time between when the primary URS shuts down and the backup URS starts up, this option minimizes the gap of acknowledging interactions pending on a routing point. This enables both the primary and backup URSs to have a synchronized view of the number of interactions pending on a routing point waiting to be distributed.

Important Information

To use the routing high availability functionality (as described in the *Universal Routing 8.1 Deployment Guide*), you need a high availability license. You set this with the licensing feature `router_ha_option` as described in *Genesys Licensing Guide*.

High availability for routing includes the ability to run with a Hot Standby redundancy type as well as the ability to set the URS option `pickup_calls` to `true`. In case of the absence of a high availability license:

- URS will downgrade the Hot Standby redundancy type to Warm Standby and force the `pickup_calls` option value to be `false`.
- The following message appears in the log:
`Std 07100 Licensing Violation...`
`Router High Availability Mode is turned off`

In summary, URS high availability mode means the redundancy type is Hot Standby and option `pickup_calls` is supported.

Warning! Do not set this option to `true` for eServices routing points. These routing points can potentially have thousands of pending requests and adversely affect the performance of URS.

For additional information, see the *Universal Routing 8.1 Deployment Guide*, “System Availability and Redundancy” chapter.

prestrategy

Location in Configuration Layer by precedence: routing point, T-Server, tenant, URS

Default value: none, by default this option is simply absent

Valid values: name of the subroutine (with no parameters) to be executed before the main strategy starts

Warning! URS does not validate that the option value is the name of some subroutine with no parameters. It is your responsibility to enter a valid subroutine name. If the specified value for the subroutine name does not meet the above criteria (for example, if the name refers to a strategy but not a subroutine or it has parameters), this can cause URS to freeze or other unexpected behavior.

Note: Beginning with release 8.1.0, URS during run time will examine each invoked subroutine to check whether the invoked script really is a subroutine. If not, it generates the error "`0018 Unknown object in strategy.`" IRD inserts code in the resulting strategy to check the value of the prestrategy option. If it is the same as the current strategy, then the subroutine is not invoked. This step serves as additional protection in case some direct manipulation of scripts in Configuration Manager changes the script type.

Value changes: take effect immediately

Every 7.2 and later strategy checks the current routing point for the presence of the prestrategy option. If specified, URS consider its value as name of subroutine without parameters that will be executed for every call on this routing point. You can use this option to give consult calls time to merge with the original call. Also see “[route_consult_call](#)” on [page 664](#).

pulse_time

Location in Configuration Layer by Precedence: URS

Default Value: 2

Valid Values: 1, 2, 3, 4

Value changes: take effect immediately

This option specifies the approximate time (in seconds) between consecutive re-checking of target states.

In order to synchronize the checking process among multiple URSs, the start time of each re-checking process is strictly defined. Each URS calculates this time based on the value of the `pulse_time` option. Checking happens as close as possible to this moment of time according to the clock of the computer running URS.

For example, assume URS starts at 10:36:13.356 a.m.

If `pulse_time` is 1, URS checks target states as close as possible (local computer time) to 10:36:14.000, then at 10:36:15.000, and so on.

If `pulse_time` is 3, URS checks target states as close as possible to 10:36:15.000, then at 10:36:18.000, and so on.

Please be aware that if synchronization between URSs is required, the value of option `pulse_time` for every URS should be the same.

Note: Because `pulse_time` controls how frequently URS checks targets ready to accept a waiting call, setting this option to 3 or 4 can reduce URS sensitivity for detecting ready “call/target” pairs. Because of this, Genesys recommends not setting `pulse_time` any larger than 2 for routing scenarios that may include the following: use of the `verification_time` option (see [page 686](#)), unmonitored IVRs, and short target waiting times in strategies. The larger values can be useful in cases of threshold-based routing since they alleviate any threshold precision issues related to network delays.

Also see “Busy Treatments in Routing Objects” on [page 366](#).

reconnect_time

Location in Configuration Layer by precedence: URS

Default value: 5

Valid value: any positive integer

Value changes: take effect immediately

Defines a time interval, in seconds, during which URS waits before attempting to reconnect to Configuration Server after losing its connection. The reconnection time used by URS for other servers is specified in the `ServerInfo` tab of the properties dialog box for the other servers.

reduced

Note: The `validate` option, which controlled verification of DNs in URS 6.5.x, no longer performs this function in URS 7.x (see [page 683](#) for more information). In URS 7.x, in order to successfully perform skill-based and/or agent group-based routing on Aspect or Spectrum switches using the agent's Login ID as the destination, the `reduced` option must be set to a value of 16, which cancels only the verification of DN existence in the Switch configuration. Should you wish to cancel an additional verification besides DN existence, then the value of the `reduced` option must be the sum of 16 + (any another valid value). See option “`validate`” on [page 684](#) for important information on DN verification in URS 7.x versus URS 6.5.x.

Location in Configuration Layer by precedence: URS

Default value: `false`

Valid values: `true`, `false`, a positive integer

Value changes: take effect immediately

Switching off features that are not used can increase the efficiency of URS. This option instructs URS not to perform certain verifications that are required only in particular circumstances. A value of `false` instructs URS to conduct all

the following verifications, while a value of `true` means there will be no verification. A numerical value switches off the verifications that correspond to that value.

The verifications that can be switched off by this option are:

- Checking for variables used in expressions inside the Skill object of the List Target (value=2).
- Checking the link between the T-Server and the switch before deciding on the availability of a target of type `.Q`, `.RP`, or `.DL` (value=4).
- Checking that the Network Destination selected from a DN Group as a result of translation with `[DN.DL]` belongs indeed to the remote switch to which the interaction should be sent. This verification only applies to Network Routing (value=8).
- Checking whether the DN reported by Stat Server is configured and enabled in Configuration Layer (value=16).

Note: To perform agent or agent place-based routing (where `Person`, `Agent Group`, `Place`, `Place Group`, or `Skill Expression` is selected) on Aspect and Spectrum switches, this flag must be set.

- Checking the reservation state of the target (value=32).
- Attempting to route an interaction when `EventTreatmentApplied` arrives (value=256).
- Not updating the content of the skills groups due to any modification made on the Annex tab of the Agent or Place configuration objects (value=4096).
- Not searching for the best (by statistics) agents in CPU saving mode (value=2048).

Note: To switch off one of the verifications, enter the corresponding value. To switch off more than one verification, add the values together and then enter the total value. For example, a value of `10` means that URS will neither verify whether targets are configured, nor whether variables are used in skill expressions, nor yet whether selected Network Destinations belong to the required switch, but it will check the link between T-Server and switch before routing to targets of type `.Q`, `.RP`, or `.DL`.

reg_attempts

Location in Configuration Layer by precedence: URS

Default value: 1

Valid value: any positive integer

Value changes: take effect immediately

Specifies the number of attempts that URS makes to register a routing point.

reg_delay

Location in Configuration Layer by precedence: URS

Default value: 2 seconds

Valid value: any positive integer

Value changes: take effect immediately

When URS is notified from Configuration Server that a new DN is being created, URS usually delays registration on this DN to give T-Server time to handle this notification first; otherwise, T-Server will reject an attempt to register. This option specifies the time URS delays registration.

report_reasons

Location in Configuration Layer by precedence: URS

Default value: `false`

Valid value: `true`, `false`

Value changes: take effect immediately

Starting from 7.6, URS uses `AttributeReason` (see “Example AttributeReason Messages” on [page 656](#)) of routing requests to automatically provide additional information for reporting purposes.

AttributeReason

This information in `Attribute Reason` includes:

1. The DBID of the URS application that routed the interaction. This data is passed as an integer value with key RTR. The primary intent is to distinguish a URS routing attempt from default routing performed by the Switch.
 - The presence of the application DBID in `AttributeReason` indicates that particular routing attempt was performed by a specific URS node.
 - The absence of this attribute indicates that URS was not involved in the routing attempt, but default routing by the Switch was performed instead.

Note: If default routing by the Switch occurs, attached data (such as reporting data) can become outdated. This can occur for all types of reporting based on interaction attached data.

2. The DBID of the strategy that URS used to route the call.
3. The same reporting information that URS attaches to the interaction when option `report_targets` (see [page 660](#)) set to true. Example: `RTargetTypeSelected`, and other keys, as described in Table 139 on [page 662](#).

As a result:

- URS always provides DBIDs in `AttributeReason` of routing requests.
- Reporting information about the selected target can be provided in the User Data of an interaction and/or in `AttributeReason` of routing requests.

Options `report_targets` vs. `report_reasons`

- Option `report_targets` (if set to `true`) facilitates the presence of reporting information in the User Data of an interaction.
- Option `report_reasons` facilitates the presence of reporting information in `AttributeReason` of routing requests.
- No reporting information is provided if both these options are set to `false`.
- If both of them are set to `true`, reporting information is provided in both the User Data and in `AttributeReason` of routing requests.
- In summary, `report_targets` and `report_reasons` control whether (`=true`) or not (`=false`) URS provides reporting data with the call. The reporting data is the same. The only difference is where reporting data will be placed. Option `report_targets` places data in call User Data and `report_reasons` places the reporting data into `AttributeReason` of the routing request. These options works independently. If both are `true` the reporting data will be provided in both places.

Note: The `report_reasons` option applies only to the final routing request (it is not applicable to IVRs, for example).

Example `AttributeReason` Messages

The field `AttributeReason` always exists; it is not optional.

From URS to T-Server

An example `RequestRouteCall` message from URS to T-Server with the `AttributeReason` highlighted in red is shown below.

```
10:56:57.217_T_I_006b017a7cecf021 [14:19] send to tserver vit_ts2
RequestRouteCall to dn 8102 on (dnis=)
request to 65200(--) message RequestRouteCall
    AttributeReferenceID      31
    AttributeReason          [14] 00 01 01 00..'RTR'      694
    AttributeRouteType        1 (RouteTypeDefault)
    AttributeExtensions       [56] 00 03 00 00..
                                'DEFAULT#'           '8102'
                                'CUSTOMER_ID'         'Vit'
                                'SWITCH'              'vit_sw2'
    AttributeOtherDN          '8102'
```

```
AttributeConnID 006b017a7cecf021
AttributeThisDN '2202'
```

From T-Server to URS

An example EventRouteUsed Message from T-Server with the AttributeReason highlighted in red is shown below.

```
received from 65200(--)172.21.1.88:3010(fd=2452) message
EventRouteUsed
    AttributeCallType          2
    AttributeCallID           43
    AttributeConnID 006b017a7cecf021
    AttributeUserData [453] 00 14 00 00..
    'RVQID' ''
    'RTargetTypeSelected'   '100'
    'RTargetRuleSelected'  ''
    'RTargetObjectSelected' ''
    'RTargetObjSelDBID' ''
    'RTargetAgentSelected' ''
    'RTargetPlaceSelected' ''
    'RTenant' 'Vit'
    'RStrategyName'        'XML'
    'RStrategyDBID' '3229'
    'CBR-actual_volume' ''
    'CBR-Interaction_cost' ''
    'CBR-contract_DBIDs' ''
    'CBR-IT-path_DBIDs' ''
    'RRequestedSkillCombination' ''
    'RRequestedSkills'(list)
    'CustomerSegment'      'default'
    'ServiceType'          'default'
    'ServiceObjective'    ''
    'PegDEF'               1
    AttributeDNIS     '800'
    AttributeANI       '11111111111111'
    AttributeCustomerID 'Vit'
    AttributeThisDN '2202'
    AttributeThisDNRole    2
    AttributeThirdPartyDN '8102'
    AttributeThirdPartyDNRole 2
    AttributeThisQueue     '2202'
    AttributeOtherQueue    '2202'
    AttributeCallState      0
    AttributeReason        [14] 00 01 01 00.. 'RTR' 694
    AttributeReferenceID   31
    AttributeTimeinSecs    1187200626 (10:57:06)
    AttributeTimeinuSecs   253000
```

report_statistics

Location in Configuration Layer by precedence: URS

Default value: `false`

Valid value: `true, false`

Value changes: take effect immediately

The `report_statistics` option can instruct URS to attach additional reporting data to calls. It supports reporting on load balancing between targets and allows for both real-time and historical reporting of calls in transition (from the network to the contact center site (to `ACDQueue`, `AgentGroup`, or `Agent targets`) from all resources. If this option is set to `true`, URS extends the reporting data with load balancing statistical information (based on the load balancing statistic you select in the Routing Selection object properties dialog box).

Note: Since the option of attaching reporting data is controlled with option `report_targets`, you must set both these options to `true` in order to have the load balancing statistical information attached to a call.

The data attached to a call when this option is set to `true` depends on the load balancing statistic selected, all of which are fully described in the chapter on load balancing in the *Universal Routing 8.1 Deployment Guide*.

For every target evaluated, URS attaches separate pieces of User Data with key `LBR_DEST`. In the following example, URS evaluated two targets (two `LBR_DEST` keys appear in the `UserData`)—two ACD queues based on `RStatLoadBalance`. Key-value pair '`RTargetObjSelDBID`' '`1008`' reflects the database identifier of the selected target.

```
AttributeUserData [632] 00 18 00 00..
  'RVQID' 'LHPGMTR7DD6038D9PU92GE4P6800000V'
  'LBR_ORIG' '122:107'
  'LBR_TS' '483305730'
  'LBR_SNUM' '10'
  'RTargetTypeSelected' '5'
  'RTargetRuleSelected' ''
  'RTargetObjectSelected' '8000_Additional_Switch'
  'RTargetObjSelDBID' '1008'
  'RTargetAgentSelected' ''
  'RTargetPlaceSelected' ''
  'RTenant' 'oxana'
  'RStrategyName' 'CallsINTransition2'
  'RStrategyDBID' '2337'
  'CBR-actual_volume' ''
  'CBR-Interaction_cost' ''
  'CBR-contract_DBIDs' ''
  'CBR-IT-path_DBIDs' ''
  'RRequestedSkillCombination' ''
  'RRequestedSkills'(list)
  'CustomerSegment' 'default'
```

```

'ServiceType'      'default'
'ServiceObjective' ''
'LBR_DEST'         '5199:-1:-1:-0.1:3.08'
'LBR_DEST'         '1008:-1:-1:-0.2:-0.2'
AttributeDNIS     '200'
AttributeCustomerID 'oxana'
AttributeNetworkDestDN 'Additional_Switch::8000'

```

Note: The five values in key LBR_DEST are separated by a colon (“:”).

Table 137 describes all five positions in LBR_DEST for each load balancing statistic described in the *Universal Routing 8.1 Deployment Guide*.

Table 137: Attached Data for report_statistics Option

Statistic	Format of data
RStatLoadBalance	1. DBID of destination
	2. -1
	3. -1
	4. Current value of StatLoadBalance
	5. Current value of RStatLoadBalance
RStatCallsInQueue	1. DBID of destination
	2. -1
RStatCallsInQueue (continued)	3. -1
	4. Current value of StatcallsInQueue
	5. Current value of RStatCallsInQueue
RStatExpectedLBEWTLAA	1. DBID of destination
	2. Number of effective calls in transit area rounded to nearest integer
	3. Average delay in transit area
	4. StatLoadBalance or Longest Available Agent for first ready agent
	5. Current value of RStatExpectedLBEWTLAA

Table 137: Attached Data for report_statistics Option (Continued)

Statistic	Format of data
RStatExpectedLoadBalance	1. DBID of destination
	2. Number of effective calls in transit area rounded to nearest integer
	3. Average delay in transit area
	4. Current value of StatLoadBalance
	5. Current value of RStatExpectedLoadBalance
RStatLBEWTLAA	1. DBID of destination
	2. -1
	3. -1
	4. StatLoadBalance or LAA for first ready agent
	5. Current value of RStatLBEWTLAA
All Other Load Balancing Statistics	1. DBID of destination
	2. -1
	3. -1
All Other Load Balancing Statistics (continued)	4. -1
	5. Current value of used statistic

Note: If evaluation of target was not performed (for example, if the target was not ready) or if the statistic cannot be evaluated (for example, if the statistic for the target is not open), then nothing is attached for such a target.

Starting with 7.6, URS also extends currently used key `RTargetObjectSelected` with its peer key `RTargetObjSelDBID`. This latter key (just like `RTargetObjectSelected`) is not controlled by the `report_statistics` option, but by option `report_targets` only (see below). Its value is the DBID of the selected target.

report_targets

Location in Configuration Layer by precedence: URS

Default value: `true`

Valid value: `true`, `false`

Value changes: take effect immediately

The intent of this option is to add information to interactions regarding targets waited for and routed to. This information can then be used for reporting.

Note: When a cost-based routing solution is implemented as described in the *Universal Routing 7.6 Cost-Based Routing Configuration Guide*, URS attaches cost-based routing reporting information to interactions when option `report_target` is set to `true` and URS routes the interaction (not the Switch as in default routing).

In 6.x, URS operated much like a black box; in other words, without exposing routing decision information.

In 7.x, routing decision data is exposed. Decision data is generated per interaction and includes:

- Routing rule, agent/place group, skill expression that an interaction is waiting on and routed to.
- Agent/place an interaction is routed to.
- Tenant, Strategy that an interaction is routed by.

To expose the routing decision data, set option `report_targets` to `true`.

When Set to True

If the `report_targets` option is set to `true`, every time an interaction enters a target object, URS attaches information about high-level targets for which the interaction is currently waiting (`AttributeUserData`). The high level targets include routing rules, skill expressions, agent groups, place groups).

Note: If the routing strategy specifies an agent list or a single agent as a target (except for a routing rule), URS does not attach any waiting key-value pair to the interaction.

When the interaction is finally routed, URS attaches information about the high-level target object and removes the information about targets for which the interaction was currently waiting.

Information Generated While Interaction Waits for a Target

The following keys are used while the interaction is waiting for a target (see [Table 138](#) and [Table 139](#)):

Table 138: Waiting for Target Information

Key	Value (string)
RTargetRule	Name of the routing rule causing the interaction to wait.
RTargetAgentGroup	Name of the agent group and/or skill expression for which the interaction is waiting.
RTargetPlaceGroup	Name of the place group for which the interaction is waiting.

Information Generated When Interaction Finally Routed

The following keys are used when the interaction is routed:

Table 139: Routed to Target Information

Key	Value (string)
RTargetRuleSelected	Name of the routing rule used to route the interaction.
RTargetObjectSelected	Name of the high-level target object the interaction was routed to (e.g., name of an agent group, agent Login ID if target is an agent).
RTargetObjSelDBID	Database identifier of selected target.
RTargetTypeSelected	High-level target type the interaction was routed to.
RTargetAgentSelected	Login ID of the agent the interaction was routed to.
RTargetPlaceSelected	Place name the interaction was routed to.
RTenant	Name of the tenant the strategy belongs to.
RStrategyName	Strategy name.
RStrategyDBID	Database identifier of the strategy that routed the call.

Example Strategy and Resulting Data Attached to Interaction

Figure 249 shows part of an example strategy where (1) an interaction enters the first target selection object, (2) the same interaction enters the second target selection object after a timeout, and (3) the resulting data generated when the interaction is finally routed.

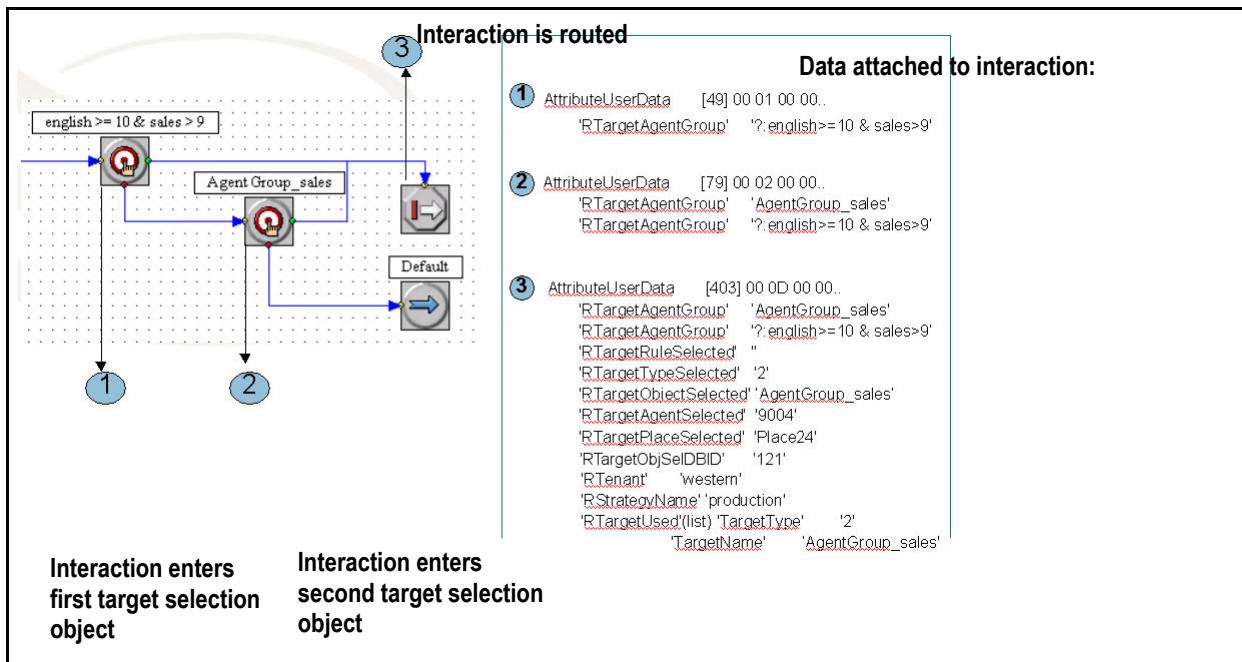


Figure 249: Option Report_Targets, Data Attached to Interaction

For information on T-Server event messages that contain user data, see the *Genesys Events and Models Reference Manual*.

Note: Because such reporting is very resource-consuming, this functionality is disabled by default.

Option `report_targets` and `report_reasons` control whether (=true) or not (=false) URS provides reporting data with the call. The reporting data is the same. The only difference is where reporting data will be placed. Option `report_targets` places data in call User Data and `report_reasons` places the reporting data into The Reasons attribute of the routing request. These options works independently.

request_timeout

Note: This option can be overridden by the SetCallOption function. See [page 475](#) for more information.

Location in Configuration Layer by precedence: Server (DB Server, Customer Server, or Stat Server), URS

Default value: `0` milliseconds

Valid value: any positive integer

Value changes: take effect immediately but don't affect interactions already waiting for a response from the server.

This option, entered in the `default` section, is used to limit the time URS waits for a response from DB Server, Custom Server, or Stat Server or Web Services. If URS receives no response, URS responds as if an error event was received from the server and the interaction is routed to the Red port on the object with a `0013` error code (Remote error). The value of this option is in milliseconds and can be specified on DB Server, Custom Server, Stat Server, and URS.

reservation_pulling_time

Location in Configuration Layer by precedence: T-Server, URS

Default value: `0`

Valid values: non-negative integers

Value changes: take effect immediately

This option specifies a time period (milliseconds) that URS expects a single reservation request can take. If the execution of `SelectDN` function (target selection object with `0` waiting time) results in sending an agent reservation request, then URS delays strategy executions and the start of busy treatments up to this interval. The purpose of the delays is to avoid the starting of any activity that could prevent routing the call to the agent, if the agent happens to be successfully reserved.

Special Note

Upon entering a target selection object with a waiting time set to `0`, in some cases URS still can wait a short time before quitting the object:

- If URS has no qualified target reported by Stat Server as ready and there is at least one target with a not-yet-opened statistic, then value of the `treatment_delay_time` option will be used as additional waiting time.
- If URS has a qualified target reported by Stat Server and issues an agent reservation request for this target (which can happen if agent reservation is enabled), then URS will wait for the maximum of values `treatment_delay_time` and `reservation_pulling_time`.

route_consult_call

Location in Configuration Layer by precedence: routing point, T-Server, tenant, URS

Default value: `false`



Valid values: `true`, `false`, any integer in milliseconds.

Value changes: take effect immediately

The functionality of `route_consult_call` provides the ability to delay routing consult calls. The valid values for this option now include time (in milliseconds) in addition to the previously-existing values of `false` and `true`. For example, specifying a value of `10000` causes a delay of up to 10 seconds.

Use the `route_consult_call` option to specify whether URS should route a transferred call (consult call type) based on the transfer start (value = `true`) or on the transfer completion (value = `false`). Time in milliseconds specifies the maximum wait time for a transfer to be finished before it starts a strategy for a consult call type. If the transfer is completed before the specified timeout has expired, URS stops waiting and begins execution of the strategy.

If an agent wants to transfer an interaction using a two-step transfer, the agent initiates an additional interaction, called a consult call. This consult call is used to call another agent directly or call a routing point on which a strategy is loaded for finding a new agent (in case no specific agent is desired). Before the transfer of the original interaction is completed, the original interaction is merged with the consult call.

The setting of the option `route_consult_call` determines whether URS should delay running the strategy for the consult call at the routing point until after the original interaction is merged with the consult call or run the strategy before the merge. Setting the option to `true` instructs URS to immediately search for an available agent when the first agent initiates a consult call to a routing point. When an available agent is found, the first agent consults with the new agent and then completes the transfer of the original interaction (at which time the original interaction is merged with the consult call). Setting the option to `false` instructs URS to begin running the strategy only when the transfer is completed (meaning that the original interaction and consult call are merged) to the routing point.

Note: The setting of `route_consult_call` does not matter if the agent is transferring an interaction directly to another agent rather than to a routing point.

For example, with the option set to `true`, if AgentA receives a call and wants to do a consult call with another agent, AgentA initiates a consult call to a routing point on which a strategy is loaded. URS runs the strategy to locate an available agent and finds AgentB. AgentA is then able to talk with AgentB. At the end of the consult, AgentA presses the button to complete the transfer, which instructs URS to complete the transfer to AgentB. The customer call is merged with the consult call and AgentB establishes a connection with the call. AgentA hangs up.

With the option set to `false`, when AgentA initiates a consult call to the routing point, URS will not begin searching for another agent until the transfer is

complete and the customer interaction is merged with the consult call. In this case, AgentA does not know which agent receives the call.

Note: Not all switches or T-Servers support this option. On some switches (or on some switch settings), transfers cannot be completed on routing points, but on agent DNs only. This type of switch will wait until the call is routed to complete the transfer and URS will wait for the switch to complete the transfer before routing the call. If this is the case for consult calls, `route_consult_call` must be set to `true` to avoid a deadlock. Setting the option to `false` deadlocks routing because URS will wait to run the strategy until after the original interaction and consult call is merged, but a regular DN won't allow a transfer to be completed until URS routes the interaction to the regular DN.

run_time_mode

Location in Configuration Layer by precedence: URS

Default value: `0` (zero)

Valid values: `16`, `64`, the sum of `16` and `64`, `128`

Value changes: take effect after restart

This option allows URS to adjust its run time behavior:

A value of `16` disables the automatic distribution of `EventDiverted` to Virtual Queues when a call is successfully routed. `EventDiverted` will be postponed until its distribution is explicitly requested in the strategy (`ClearTargets()` function) or the call is deleted from URS memory.

A value of `64` disables the entering of Target Selection objects/functions immediately after `EventReleased`, or if `EventAbandoned` is received for an interaction.

To activate both adjustments to the functionality - the sum of these values may be used (`80`).

When a value of `128` is set, URS considers the dot (.) character as a regular character while operating with List Objects. When the `run_time_mode` option is set to its default value of `0` (zero), more precisely the 8th bit of the value is clear, URS uses the dot (.) character as a separator between list/sublist key names.

A value of `0` disables these adjustments.

service_timeout

Location in Configuration Layer by precedence: URS

Default value: `30` (for compatibility with 6.x URS where this value is hardcoded)

Valid values: any non-negative integer

Value changes: take effect immediately

Specifies the time in seconds that URS will wait for a response from an external service, such as a service from Interaction Server. This option also applies to ICS 6.x Acknowledgement (see “6.x Compatible Operation Mode” on [page 176](#)) and Autoresponse strategy objects (see “6.x Compatible Operation Mode” on [page 189](#)).

skip_targets

Location in Configuration Layer by precedence: URS

Default value: `empty`

Valid values: `empty`, `logout`, `never`

Value changes: take effect immediately

Use this option to have URS skip waiting time if all agent targets are logged out and avoid the performance impact of expanding agent groups and scanning all agents in order to check their login status in the strategy.

- A value of `empty` instructs to skip waiting time if all agent targets are empty. For example, a target is specified a variable and this variable has an empty string as its value (such as `x=ExpandGroup[Group1]` and `Group1` is empty).

Note: A value of `empty` does not cause skipping of the target when you target an Agent Group with no one logged in. In case of an “empty” target, there is no such agent so URS does not wait. You must specify an option value of `logout` in order to skip the wait time.

- A value of `logout` instructs URS to skip waiting time if there is not at least one logged in agent. This value avoids having a strategy wait in a target Selection object or all agents already waiting for the call in order to determine if all agents are logged out. Prior to this option, designers needed to create a strategy with an explicit loop that processed every agent from the target set and requested the logout status for each agent; such as `SData[StatLoggedOut]`. If no one was logged in, the strategy did not go inside corresponding target Selection object. Using the `logout` option value helps to avoid such an explicit loop in a strategy and works much faster.
- A value of `never` instructs to never skip waiting time.

startup_verbose

Location in Configuration Layer by precedence: URS

Default value: current value of option `verbose`

Valid values: from `0` to `5`

Value changes: take effect on restart

URS uses this value for the log output during the URS startup and initialization. After initialization is completed, the log level is set according to the value specified in the option `verbose`. This allows reduced URS logging on start up, making URS ready to process calls as quickly as possible.

strategy

Location in Configuration Layer by precedence: T-Server, Tenant, URS
 Default value: `none`

Valid values: the name of strategy that URS must execute when interaction arrives at a routing point

Value changes: take effect immediately

You can use this option to run the Genesys-supplied strategy described below, which routes outbound interactions to Campaign Groups.

'To support multi-Campaign agent management, the URS installation package includes pre-written strategy bytecode (`OutboundMultiCampaign.ooo`) that routes outbound interactions to Campaign Groups.'

If you are a pure Genesys Outbound Contact customer, you do not have the rights to edit strategies. In this case, to activate automatic outbound routing:

1. Load `OutboundMultiCampaign.ooo` file into URS memory as described in the *Universal Routing 8.1 Deployment Guide*.
2. Instruct URS to run `OutboundMultiCampaign.ooo` strategy for every unloaded routing point by setting the URS option `strategy` to value: `OutboundMultiCampaign`

Note: For information on the new `Campaign Group` target in the Routing Selection and Route Interaction strategy objects, see the section on Statistical Objects (Target Types) in the Interaction Routing Designer Objects Chapter of the *Universal Routing 8.1 Reference Manual*.

targets_order

Location in Configuration Layer by precedence: URS

Default value: `random`

Valid values: `random`, `fifo`

Value changes: take effect immediately

This option controls the mechanism for the default target selection. If this option has a value of `random`, URS will pick a target from the list according to its internal rules when it encounters a target list with more than one available target. When this option is set to `fifo` and URS tries to route an interaction to a target list where more than one target with the same statistical value is available, it will route the interaction to the first available target in the list.

Note: StatAgentLoading (see [page 722](#)) will always use random target order despite the setting of `targets_order`.

tattributes

Location in Configuration Layer by precedence: URS
 Default value: empty, no snapshot is taken of any attribute.
 Valid values: a value that contains a comma-separated list of attribute numbers
 Value changes: takes effect after restart
 A snapshot is taken of only the attributes that are listed as values of this option and accessed by the `GetRawAttribute` function.

Threshold

Location: Annex properties of DN controlled by URS or the DN Group specified in the option `ThresholdGroup`, in a folder named `ParkingThreshold`
 Default value: none; a value must be supplied when the parking threshold mechanism is used
 Valid values: positive integers
 Value changes: take effect immediately
 The upper limit of interactions for which URS can apply treatments. All additional interactions will be sent to some default destination specified with the following two options: `ThresholdDestination` and `ThresholdSwitch`.

Note: This option is case sensitive.

ThresholdDestination

Location: Annex properties of DN controlled by URS or of the DN Group specified in the option `ThresholdGroup`, in a folder named `ParkingThreshold`
 Default value: none; if no value is supplied, the configured default destination will be used
 Valid values: configured DNs (string)
 Value changes: take effect immediately
 Interactions over the limit will be sent to this DN number.

Note: This option is case sensitive.

ThresholdGroup

Location: Annex properties of DN controlled by URS, in a folder named `ParkingThreshold`

Default value: none; if this option is not specified, the threshold limit will be based on this DN alone

Valid values: configured DN groups (string)

Value changes: take effect immediately

This option is used to impose limitations to groups of routing points so they will share a common threshold for incoming interactions. Interactions will be sent to `ThresholdDestination` if the sum of waiting interactions exceeds the limit set in the `Threshold` option.

In addition to these options, also define a filter named `CallParked`, either as an interaction object in Configuration Layer or as a configuration option in Stat Server. This filter's definition must be `PairExists["CallParked", 1]`.

The parking threshold mechanism works as follows: when an interaction comes to a routing point (virtual routing point, service number) for which the option `use_parking_threshold` has a value of `true`, URS requests from Stat Server the value of the statistic `CurrNumberWaitingCalls` filtered by `CallParked`, for the routing point (if `ThresholdGroup` is not specified) or for the specified DN Group. If this statistical value exceeds the specified threshold, the interaction is sent to:

- The default destination if `ThresholdDestination` is not specified alongside `Threshold`.
- The DN given by `ThresholdDestination`, on the same switch as the routing point, if `ThresholdDestination` is specified but `ThresholdSwitch` is not.
- The DN given by `ThresholdDestination` on the switch given by `ThresholdSwitch` if both options are specified.

Otherwise, URS starts the loaded strategy for the interaction.

When the option `use_parking_threshold` has a value of `true`, a key-value pair with key `CallParked` and value `1` is attached to the interaction when a treatment is started for the interaction. This pair allows URS to discern the parked interactions, which are the only interactions used in computing the threshold statistic.

Important Information

- This option is case sensitive.
- If the threshold is specified for a DN instead of for a group of DNs, the DN must be given an Alias in Configuration Layer. The options `Threshold`, `ThresholdDestination`, and `ThresholdSwitch` must be specified in the same location (either in the properties of the DN or in those of the Group of DNs).

ThresholdSwitch

Location: Annex properties of DN controlled by URS or of the DN Group specified in the option `ThresholdGroup`, in a folder named `ParkingThreshold`

Default value: the same switch as that of the DN from where the interaction is routed
Valid values: configured switch names (string)
Value changes: take effect immediately
The switch name where the `ThresholdDestination` DN is located.

Note: This option is case sensitive.

transfer_time

Location in Configuration Layer by precedence: URS
Default value: Current value of `transition_time` option (default value of `transition_time` option is 15)
Valid value: any positive integer
Value changes: take effect immediately
Helps URS to spot lost interactions. Some events from T-Server can indicate that an interaction is gone (`EventRouteUsed` or `EventReleased`) but not necessarily every time. URS marks every questionable interaction. During `transfer_time`, if there are not any events signaling that the interaction is still alive, the interaction is assumed to be gone and all information about this interaction is deleted from URS memory. Before this option, the URS used the `transition_time` option instead. If `transfer_time` is not specified, however, URS will return to using `transition_time` for this purpose.

transfer_to_agent

Location in Configuration Layer by precedence: T-Server, URS
Default value: `false`
Valid values: `true`, `false`
Value changes: take effect immediately
Instructs URS to request T-Server to transfer interactions from an IVR directly to a target agent instead of returning them to the routing point. Use where standard routing scenarios do not apply or cannot be used to override the usual method of routing a call to an agent. For example, there may be special hardware or reporting needs. Allows you to initiate a direct transfer to an agent using T-Library functionality. For more information on T-Library functions, see the T-Library Functions section of *Genesys Voice Platform SDK .NET 7.5 API Reference*.

transit_dn

Location in Configuration Layer by precedence: any DN that URS does not register, either by default or because of some other option
Default value: none

Valid values: `true`, `false`

Value changes: take effect immediately

When set to `true`, instructs URS to register for the purpose of monitoring calls at the DN. When this option is set to `false` on a Virtual Queue, it prevents URS from registering on the Virtual Queue. As a result, T-Server does not distribute any events in respect of this Virtual Queue to URS. URS continues to distribute (more precisely asks T-Server to distribute) Virtual Queue events, even if it is not registered to T-Server.

transition_time

Location in Configuration Layer by precedence: routing point, T-Server, URS

Default value: 15

Valid value: any positive integer

Value changes: take effect immediately

Specifies the minimal time, in seconds, for which URS will wait between the moment an interaction is routed to an agent, a place, or a DN, and any subsequent check for routing to the same agent, place, or DN. A nonzero (3 or higher) value is needed to avoid repeated routing to the same agent or place before Stat Server has reported the agent or place as busy. The value of this option should be increased when interaction traffic and network traffic intensify. The default value of 15 should ensure enough time for Stat Server to report an agent's change of status.

Note: This option is also used for agent reservation. See the description of “agent_reservation” on [page 629](#). For additional information on how the `transition_time` value is used during agent reservation, see the *Universal Routing 8.1 Deployment Guide*, “System Availability and Redundancy” section.

Important Information

- Although any positive integer is a valid value, the value should be 3 or higher to prevent URS from selecting the same target for different interactions.
- When testing strategies offline, the `transition_time` option can be set to a lower value, such as 5.

treatment_delay_time

Location in Configuration Layer by precedence: T-Server, URS

Default value: 0

Valid values: non-negative integers

Value changes: take effect immediately

Note: When URS processes the first interaction after being started and option `treatment_delay_time` is used, URS adjusts wait time to the maximum allowed by the option. As a result, the first interaction no longer has the potential of being abandoned when this option is used.

This option may prove useful when the agent reservation feature is used because of the delay caused by the reservation exchange between URS and T-Server. It gives Stat Server time to provide URS with information about targets, preventing URS from applying busy treatments before determining whether a target is available. A nonzero value, specified in milliseconds, delays the start of the first busy treatment for every Routing object that uses busy treatments. The interval should be short enough so the caller will not notice a delay, but should allow enough time for the agent availability information to be retrieved from Stat Server. A value of `0` means that busy treatments may occasionally be applied even though a target is available.

For additional information on `agent_reservation` option, see the *Universal Routing 8.1 Deployment Guide*, “System Availability and Redundancy” chapter.

See also: “`reservation_pulling_time`” on [page 664](#).

Note: Upon entering in any target selection object, if a target was not selected and at least one statistic is not open and waiting time is `0`, URS adjusts waiting time to the maximum of `reservation_pulling_time` and `treatment_delay_time` values.

unix_socket_path

Location in Configuration Layer by precedence: Stat Server

Default value: `none`, a value must be supplied if a socket path will be used

Valid value: any valid socket path to the Stat Server host

Value changes: take effect on restart

Specifies a socket path to use to connect to the Stat Server host. This is an alternative way to specify the host name in some UNIX systems.

unloaded_cdn

Location in Configuration Layer by precedence: routing point, T-Server, tenant, URS

Default value: `defaultignore`

Valid values: `ignore`, `reject`, `default`, `defaultignore`

Value changes: take effect immediately

Specifies what action to take if an interaction comes into a DN without any strategies loaded. The values are defined as follows:

- `Ignore` means that URS will take no action.
- `Reject` means that URS will send a routing instruction to T-Server with route type `Reject` (which, in general, leads T-Server to delete the interaction)
- `Default` means that the interaction will be sent to the default destination.
- `DefaultIgnore` is used for compatibility with URS version 5.1. This value means `default` if an interaction comes in to a routing point and `ignore` if an interaction comes in to an IVR. Therefore, if the option value is `defaultignore`, URS will send an interaction that arrives on a routing point to the default destination and do nothing with an interaction that arrives on an IVR.

use_agent_att

Location in Configuration Layer by precedence: Virtual Queue, URS.

Default value: `default`

Valid values: `default`, `always`.

Value changes: take effect immediately.

Defines how the option `agent_att`, [page 629](#), is used by URS for calculating some metrics returned by the function `RvqData`, [page 570](#).

If the option is set to `default`, URS uses the value of `agent_att` until the agent completes 10 calls.

If set to `always`, URS uses only the value defined in the option `agent_att`.

The options `agent_att` and `use_agent_att` are applicable for processing voice interactions only.

Note: The option `use_agent_att` is enabled if the option `agent_att` has a value greater than 0.

use_agent_capacity

Note: The URS Application template sets the `use_agent_capacity` option to `true`. Setting this option to `true` routes based on the agent capacity model as described in the *Genesys 8.1 Resource Capacity Planning Guide*. If your site does not want to route based on the agent capacity model, switch this option to `false` after importing the template.

Location in Configuration Layer by precedence: Stat Server, URS

Default value: `true`

Valid values: `true`, `false`

Value changes: take effect on restart

Starting with Universal Routing 7.0.1, URS is required to use agent capacity information supplied by Stat Server when routing non-voice interactions. For example, URS must use agent capacity information in a blended environment (such as when routing e-mail plus voice or e-mail plus chat) or when routing other media types (such as when routing only e-mail or only chat).

URS can use agent capacity information provided by Stat Server to determine an agent's ability to accept a particular interaction at a particular time. When factoring in agent capacity information, URS selects from the pool of agents with the required business skills and chooses the available agent(s) based on agent capacity information (instead of current agent state).

- If more than one agent is available, URS selects one based on a specified statistic (see “Route Interaction” on [page 322](#) and “Selection” on [page 326](#)).
- If a statistic is not specified, URS selects based on agent load (instead of the agent with maximum waiting time).

Use Case

Assume the following agent/place (desk) configuration:

- An agent is configured in Configuration Manager with a capacity rule, which allows the agent to have three voice interactions and one e-mail interaction simultaneously.
- The agent has multiple Login IDs assigned to him (assume three Login IDs: `LoginID#1`, `LoginID#2`, `LoginID#3`).
- A Place is configured in CME, which has shortcut to multiple DNs (three DNs in this use case: `DN#1`, `DN#2`, and `DN#3`).

Next, assume the following routing scenario:

- The agent logs in on multiple voice DNs using different Login ID for each login operation (on `DN#1` with `LoginID#1`; on `DN#2` with `LoginID#2`; and on `DN#3` with `LoginID#3`).

Note: It is not uncommon for a single agent to be logged in with different phones at a single place (desk). The assumption here is that the switch allows this.

- The agent receives the first call.
- The agent receives the second call (while still on the first call).
- The agent receives a third call (still has the first and second call).

Effect of Agent Capacity Rule

In this case, with the above agent capacity rule configuration, the agent cannot receive a fourth call. In order to get the fourth call, the agent has to release one of the previous calls.

The `use_agent_capacity` option, the `UseMediaType` (see [page 479](#)), and `GetMediaTypeName` (see [page 474](#)) functions, and the `StatAgentLoadingMedia` statistic (see [page 724](#)) all support the Genesys agent capacity distribution model.

Important Information

- The URS Application template sets the `use_agent_capacity` option to `true`. Genesys assumes that many new customers installing Universal Routing 7.2 and later are (or will be) routing both voice and non-voice interactions and will want to use agent capacity rules. Genesys eServices requires that this option be set to `true`.
- Routing based on agent capacity rules has benefits in voice scenarios where agents can have multiple voice DNs. Voice-only customers who do not define agent capacity rules and set `use_agent_capacity` to `false` will see no behavioral differences from previous versions of Universal Routing.
- URS switches to supporting the agent capacity model if:
 - Stat Server 7.0.1 or later is used.
 - The option `use_agent_capacity` is set to `true`.

Note: When requesting agent state information, URS can open the `CurrentState` statistics (value of option is `false`) or the `CurrenttargetState` statistics (value of option is `true`). URS is “aware,” however, that the version of Stat Server before 7.0.1 does not support the `CurrentTargetState` statistics. For this reason, URS always asks for the older `CurrentState` statistics, no matter whether the value of this option is `true` or `false`.

- The `use_agent_capacity` option can be specified on the URS or Stat Server level. URS checks the option upon connecting to Stat Server.
- Routing based on agent capacity rules can be set at the Person, Place, or Tenant level.
- The state of agent readiness in CCPulse+ and a capacity rule’s state of readiness differ. See [page 741](#) for more information.

`use_agentid`

Location in Configuration Layer by precedence: Stat Server, URS

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect immediately

When set to `true`, interactions will be routed to agents based on their Employee IDs in the Configuration Database. If set to `false`, interactions will be routed

based on agent DNs. The value of this option depends on the requirements of the T-Server for the specific switch.

Note: Option `use_agent_id` does not affect the value of the `RequestRouteCall` Extension attribute DN. The value always contains the DN number as reported by Stat Server. The real access number (old content of attribute DN) is placed in an Extension attribute called `ACCESS`.

use_dn_type

Note: The `use_dn_type` option is extended with one additional location in Configuration Layer: It can now be specified on Switches. If specified on the Switch, URS gives the Switch the maximum priority and considers it as the destination Switch.

Location in Configuration Layer by precedence: switch, routing point, T-Server, tenant, URS

Default value: `any`

Valid values: `position`, `extension`, `any`

Value changes: take effect immediately

Specifies the default type of DN to be used. A value of `position` or `extension` means that only available DNs of type ACD Position or Extension, respectively, will be used for the routing destination of an interaction. A value of `any` means that any type of available DN can be used.

Important Information

- Do not use the value `any` for eServices interactions.
- The value of this option can be explicitly overridden by invoking the `UseDNTType` function in a strategy. Invoking the `UseDNTType` function is imperative if interactions other than telephone calls (that is, e-mail or fax) will be routed and T-Server doesn't provide `MediaType` in `EventRouteRequest`.
- If URS is routing in a situation where an agent's Place is equipped with more than one DN type (or, starting with 7.0.1, if the agent's Place works with more than one media type), then set option `use_dn_type` to `extension` or `position` for routing points that distribute voice calls. As an alternative, use function `UseDNTType` in the strategy.

use_extrouter

Note: The `use_extrouter` and `use_extrouting_type` options are extended with one additional location in Configuration Layer: they can now be specified on Switches. If specified on the Switch, URS gives the Switch the maximum priority and considers it as the destination Switch. As a result, URS can now route calls from a single routing point to multiple remote switches utilizing different external routing types. Also, if URS receives an empty Access Number as an answer for request `GetRemoteAccessCode`, then URS will proceed working with the original destination number.

Location in Configuration Layer by precedence: switch, routing point, T-Server, URS

Default value: `false`

Valid values: `true`, `false`, `local`, `remote`

Value changes: take effect immediately

Instructs URS to support external routing through the Extrouter component of the target T-Server. Any value except `false` means that URS uses external routing.

In addition, the `use_extrouter` option specifies which component performs the external routing, URS or T-Server. It is preferable to have T-Server do the external routing, which is why the option is `false` by default. If for some reason, T-Server cannot do external routing (as indicated by the Genesys documentation for the specific T-Server), then URS can be forced to do the external routing by setting the option to `true`.

When this option is used, URS requests access numbers from T-Server. If the option is set to `true` or `local`, URS requests access numbers from the originating T-Server. The originating T-Server requests the access number from the target T-Server on behalf of URS. This means that URS does not need to be connected to the remote T-Server, only to the local T-Server.

If the option is set to `remote`, URS requests access numbers from the remote T-Server (where agents are located) using the time parameter from the `extrouter_timeout` option (see [page 641](#)), if configured. In this case, the remote T-Server must be in the `Connection` tab of the URS Application.

Can be overridden with the `SetCallOption` function (see [page 475](#)).

Note: External routing for T-Servers is now called Inter Server Call Control (ISCC). For information on T-Server support of external routing, consult the chapter on Multi-Site Support in any T-Server deployment guide.

use_extrouting_type

Note: This option specifies the external routing type. See the `use_extrouter` option description for how to turn on external routing by URS.

Location in Configuration Layer by precedence: switch, routing point, T-Server, URS

Default value: `config`,

Valid values: `config`, `default`, `route`, `direct`, `reroute`, `directuui`, `directani`, `directnotoken`, `dnis`, `directdigits`, `pullback`, `extprotocol`, `routeuui`, `directnetworkcallid`.

Value changes: take effect immediately

Note: Figure 250 on [page 706](#) shows an example Switch Access Code Properties dialog box where external routing types are specified for a destination switch. For a detailed description of each valid value, see the chapter on Multi-Site support in the applicable T-Server deployment guide.

Instructs URS which type of external routing to use if external routing is on (see option `use_extrouter` for more information). Can be overridden with the `SetCallOption` function (see [page 475](#)).

The default value, `config`, specifies that the type of external routing is taken from Configuration Database. URS looks for a Switch Access Code for the External routing point Target Type from the switch where the call is located to the switch where the target is located. As shown in Figure 250 on [page 706](#), the Route Type of the Switch Access Code is used to obtain external routing type that will be used. In all other cases, including when no Switch Access Code can be found, URS uses the value of `route` for the `use_extrouting_type` option.

When this option has its value set to `dnis`, it enables support for IVR Server in In-Front load balancing mode. (See the *Universal Routing 8.0 Routing Application Configuration Guide* for configuration details.) In 8.1.1 and later releases, this IVR load balancing function is enhanced to support EPN partitioning. For example, if a DN has the `epn` property in the T-Server > Annexes section, URS also checks every ISCC resource to see if it has the same value as the `epn` property.

Warning! Put option `use_extrouting_type` in the same location as option `use_extrouter`. If this is not done, URS ignores this option and uses default value of `config` to determine the type of external routing.

use_ivr_info

Note: This option can be overridden by the SetCallOption function. See [page 475](#) for more information.

Location in Configuration Layer by precedence: RP, T-Server, tenant, URS

Default value: `true`

Valid values: `true`, `false`

Value changes: take effect immediately

If set to `true`, URS starts a strategy from the beginning when an interaction is transferred to a routing point controlled by URS, unless URS itself sends the interaction to that routing point.

This feature makes sure that an incoming interaction will not be counted several times when virtual queue statistics are calculated. It does this by remembering the virtual queues when the strategy is restarted. `EventQueue` is not sent when the interaction is placed again in such a virtual queue that corresponds to a target list. `EventDiverted` is not sent for the virtual queues where the interaction had been placed when an interaction is sent back to the routing point from an IVR.

Use Cases:

- Turn this option off when IVR is used for mandatory treatment, such as when each caller is greeted with a welcome message. In this case, when IVR returns the call to the routing point, URS continues the strategy right after the mandatory treatment.
- Turn this option on only if you expect IVR to return new data to the routing strategy. In this case, URS starts the strategy from the very beginning so that the new data will be evaluated.

Warning! Under this latter scenario, the mandatory treatment might play again. If you wish to prevent this from happening, set up attached data to indicate whether the mandatory treatment has played once or not. Check the attached data at the beginning of the strategy to determine if the call is being returned from IVR (mandatory treatment should not be played) or if it is a fresh call (mandatory treatment should be played).

use_parking_threshold

Location in Configuration Layer by precedence: DN, T-Server, tenant, URS

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect immediately

This option turns the parking threshold mechanism on and off. The parking threshold is used when the number of available IVR ports is limited. When available IVR ports are limited, treatments can be applied only for a limited number of incoming interactions. If the number of interactions waiting for ready targets reaches some limit for all newly arrived interactions, treatments cannot be applied. The `use_parking_threshold` option allows URS to handle such interaction overflow.

- If option value is `false` (default), then the URS will not control the queue size of waiting interactions on the routing point that allows unlimited growth of this queue.
- If option value is `true`, then URS will use a set of other options to limit the waiting interactions queue size. These additional options are specified for routing points only or for groups of DNs related to this routing point in Annexes under the folder `ParkingThreshold`.

This option can be specified for routing points, T-Servers, tenants, and URSSs.

use_service_objective

Note: For additional information, see the section on business-priority routing in the *Universal Routing 8.0 Routing Application Configuration Guide*.

Location in Configuration Layer by precedence: URS

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect immediately

This option is used with the PriorityTuning function. It specifies whether to use Service Objective when selecting interactions for routing.

Service Objective is the time objective to service an interaction. You have the option of defining Service Objectives in Configuration Manager (see “Assigning Business Attributes” on [page 144](#)) and attaching Service Objectives to interactions (see “MultiAttach” on [page 140](#)). The complementary PriorityTuning function (see [page 603](#)) also uses Service Objective to scale the time interval that an interaction waits.

Service Objective Versus Service Level Routing

What’s the difference between the Service Factor in service level routing (see “Service Level” on [page 333](#)) and routing based on Service Objectives as described in the section on business-priority routing in *Universal Routing 8.0 Routing Application Configuration Guide*.

Answer: As detailed below, service level routing is one dimensional while routing based on Service Objectives (Service Objective in Recommended Settings table in above guide) is multi-dimensional.

Service Level Routing

The Service Factor in a service level routing rule defines how quickly interactions in a queue need to be distributed (routed). It focuses on traditional queue performance, which assumes all interactions in the queue have the same service objective and are of the same media type. Service-level routing works by monitoring whether the rate of routing interactions within the queue is meeting a particular service level objective, say 80/20, meaning 80% of calls must be routed with 20 seconds. If the service factor drops below 80/20, then the agent pool can be expanded automatically to add more agents to the queue to compensate for or prevent the loss of the defined service objective.

Service Objective

Using the capabilities defined in the section on business-priority routing in the the *Universal Routing 8.0 Routing Application Configuration Guide*, interactions in queues can have different service level objectives based on business criteria. This enables URS to look at all interactions at the head of all queues, compare the service level objectives for those interactions, and make a decision as to the interaction to route next based on the interaction with the highest risk of missing its Service Objective. The interaction with the highest risk of service objective violation is selected and sent to an agent without the need to add or expand agent resource pool.

use_translation

Location in Configuration Layer by precedence: routing point, T-Server, tenant, URS

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect immediately

Turns number translation on or off. Number translation transforms a number recognized by URS into another number, which may carry additional information, and is passed to the switch (see [Chapter 5](#)). A value of `true` means that number translation is enabled; `false` means that it is disabled. For Network Routing 6.1/6.5, set the value to true at the Network T-Server under the Annex tab. There must be at least one translation record. For Enterprise Routing, set the value to `true` if number translation is required. For additional information, see “Number Translation” on [page 705](#).

Important Information

- If the value is `true`, the switch access code must be provided.
- Even when number translation is off, URS will attach the prefix specified in switch-to-switch access codes to the beginning of the DN to which the interaction will be routed.

- In URS 7.x, skill expressions are considered as agent groups and require switch access codes for agent groups for proper translation.

using

Location in Configuration Layer by precedence: Message Server

Default value: none

Valid values: lds (load distribution server)

Value changes: take effect on restart

Note: Specify this option in the Annex tab of the Message Server Application object in the folder with the name `__ROUTER__` or with the name of the URS Application object.

Starting with 7.6, URS can use Messages Server for reasons other than logging communications with IRD. Use this option to specify that URS will use this Message Server for any type of inter-router (URS) communications (see option “lds” on [page 645](#)). The option value of `lds` results in URS using this Message Server exclusively for inter-router communications and not for logging communications with IRD.

For more information on this option, see the sections on IVR Server load balancing and Router Self-Awareness in the Load Balancing Chapter of the *Universal Routing 8.1 Deployment Guide*.

validate

Location in Configuration Layer by precedence: T-Server, URS

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect immediately

This option is only meaningful in a multi-tenant environment. It instructs URS to validate interactions. This is done as follows:

1. URS verifies that a tenant name exists that matches with the CustomerID in the interaction information sent by T-Server.
2. URS verifies that the T-Server that sent the routing request is configured as an application in Configuration Layer.
3. URS verifies that the switch controlled by T-Server is configured in Configuration Layer and that the DN from where the routing request came is configured on the switch.
4. URS verifies that at least one DN Group that contains the DN from which the routing request came belongs to the tenant specified by the CustomerID.

If the verification fails at any of the above steps, URS rejects the interaction—it sends a routing instruction to T-Server with route type `Reject`.

If this option is set to `true`, URS will ignore a DN that Stat Server reports as available to receive an interaction if this DN is absent from the Configuration Database. This verification can be turned off by setting the fifth-lowest bit (=16) of the value of the reduced option to `0`.

Important Information

- The value for this option should always be `false` in a single-tenant environment.
- In URS 6.5.x, verification was controlled with two options:
 - 1) `validate = true` (which impacted performance)
 - AND
 - 2) `reduced != 16` (or do not include 16 as a component).

If any of these conditions were not met, verification did not occur. Usually `validate` was set to `false` and, as result, URS 6.5 did not perform any verification and routed on agent login ID. When `validate = true`, URS 6.5 behaved exactly like URS 7.x with the same performance impact since verification of target DNs is a very resource-consuming operation.

In URS 7.x, performance is no longer an issue. Because verification is logically not related with option `validate` (`validate` was introduced to verify incoming calls, not target DNs), verification of DNs is detached from the `validate` option and controlled only with option `reduced`. As result URS 7.x by default verifies target DNs whereas URS 6.5 by default did not verify them.

See option “`reduced`” on [page 653](#) for more information.

verbose

Location in Configuration Layer by precedence: routing point, URS

Default value: `false`

Valid values: `0` (`false`), `1`, `2`, `3`, `4`, `5` (`true`), `6`

Value changes: take effect immediately

Like most URS options, this one is specified in `default` section of the URS Application object.

Warning! Do not confuse this option with other options having the same name, but located in different sections of this book. See “[Other Verbose Options](#)” below for details.

You can specify a verbose level (types of log messages to be written to log) separately for every routing point. If a message is associated with a call

belonging to a routing point, URS determines which messages to write to the log based on the value of the `verbose` option that you set for URS, as well as for the routing point. If you are interested only in messages associated with calls routed from a specific routing point, set URS's `verbose` level to `0`, and the routing point's `verbose` level to `5`.

Appendix A on [page 751](#) discusses `verbose` in the `default` section. Valid values can be from `0` to `6`, where `0` is equivalent to `false`, and `5` is equivalent to `true`. In a production environment, the `verbose` option in the `default` section will normally be set to `true`.

Warning! Setting the option `verbose` to values above `5` is used for debugging purposes only and can result in extreme amount of log output.

Each non-zero value causes a different set of log messages to be written to the log. The numbers are cumulative. That is, if you set the value to `2`, the log contains message types that are marked with `1` and `2`. If you set the value to `4`, the log includes all log message types except those identified as level `5`.

- Appendix A on [page 751](#) lists the log messages associated with each valid value.

In release 7.2 and later, you can change the number associated with each log message. For example, if you want to set the `verbose` option to `3`, but also want to see a message type currently marked as level `5`, you can change the message type level from `5` to `3`. This prevents you from having to set the value to `5` in order to get the level `5` message type you want and then having to sort through all the rest of the level `4` and `5` messages that you are not interested in using.

Other Verbose Options

Do not confuse this option with the other options having the same name located:

1. In the `log` section. Control Management Layer logging levels and the output type. Appendix A, “Log Events,” on [page 751](#), contains a brief overview of Management Layer logging. For complete information on the settings for the `verbose` option in the `log` section, refer to the *Framework 8.1 Deployment Guide* (Management Layer Functionality) or to the *Framework 8.1 Solution Control Interface Help*.
2. In a user-defined `web` section for use by the Web Service object or in Web API Options. Controls the HTTP Bridge verbose level. See “Web Services Options” on [page 687](#) and “Web API (URS-Behind) Options” on [page 692](#).

These `verbose` options control different things and are unrelated to the URS `verbose` option described above.

verification_mode

Location in Configuration Layer by precedence: URS

Default value: `once`

Valid value: `once`, `always`

Value changes: take effect immediately

In previous releases, URS did not reset the `verification_time` timer when an agent changed his/her status for `WaitForNextCall` while already blocked for verification time. To address this issue, URS now provides an option called `verification_mode` (specified dynamically on the URS level).

If set to `once`, URS blocks the agent for the specified verification time only if the agent is not already blocked for verification. If set to `always`, URS blocks the agent for the specified verification time without any consideration about whether or not the agent is already blocked.

verification_time

Location in Configuration Layer by precedence: URS

Default value: `0`

Valid value: any non-negative integer

Value changes: take effect immediately

A positive value is useful when it is possible for an agent to pass through ready state in a transition between ending a call and making himself unavailable by a manual command. This option specifies the time, in seconds, that must elapse after an agent ends a call before the agent is reported as ready to receive calls again.

This option is applied every time an agent becomes ready, not just after ending a call. It is not related to any previous interaction and it is not necessary to have finished a previous interaction.

When `CheckAgentState` is set to `false` or function `UseAgentState` (see [page 554](#)) is used, URS does not apply the `verification_time` option to agents, but still applies it to agent DNs.

Note: Starting with 7.5, the behavior of options `verification_time`, `verification_time_agent`, and `verification_time_dn` are slightly modified. URS continues to use them as before. However, the condition changes that URS uses to select `verification_time_agent` or `verification_time_dn`.

If, according to Stat Server, an agent becomes ready and the agent state before becoming ready (according to Stat Server) was one of: `Dialing`, `Ringing`, `OnHold`, `CallUnknown`, `CallConsult`, `CallInternal`, `CallOutbound`, `CallInbound`, then URS blocks the agent and his DNs for $\max(\text{verification_time}, \text{verification_time_dn})$. In all other cases, URS uses $\max(\text{verification_time}, \text{verification_time_agent})$.

verification_time_agent

Location in Configuration Layer by precedence: URS

Default value: 0

Valid value: any non-negative integer

Value changes: take effect immediately

While the above `verification_time` option is applied after an agent is reported as ready due to any event, `verification_time_agent` and `verification_time_dn` are applied only after specific events.

- If an agent state event is related to an agent's DN state (agent hangs up the phone, for example), then URS uses `max(verification_time, verification_time_dn)` as the verification time to block the agent.
- If an agent state event is not related to an agent's DN state, (the agent pushes the Ready button, for example), then URS uses `max(verification_time, verification_time_agent)` as the verification time to block the agent.
- Router considers the following agent state changes as DN related: `OffHook`, `CallDialing`, `CallRinging`, `CallOnHold`, `CallUnknown`, `CallConsult`, `CallInternal`, `CallOutbound`, `CallInbound`.
- See note for `verification_time` option.

verification_time_dn

Location in Configuration Layer by precedence: URS

Default value: 0

Valid value: any non-negative integer

Value changes: take effect immediately

- If an agent changes his state from the `OffHook` state to the `WaitForNextCall` state, URS will apply the `verification_time_dn` option for agent verification.

Also, see the description above for `verification_time_agent`.

Web Services Options

To communicate with Web Services through SOAP/XML and/or REST over HTTP/HTTPS protocols, URS uses a component called HTTP Bridge. You configure HTTP Bridge using various options in the URS Application object. There are options that you can use to control tracing and debugging when you use HTTP Bridge to access Web Services and to adjust HTTP Bridge performance characteristics.

For details see Appendix B, “IRD Web Service Object,” on [page 771](#), and the *Universal Routing 8.1 Deployment Guide*.

Notes:

- If using HTTP Bridge to access SAP RFC functions, you must also have installed the *Gplus* Adapter for mySAP Data Access Component.
 - Web Services functionality is also used to implicitly communicate with Workforce Web Services. Web Services functionality must be enabled if any Workforce related functionality will be used.
 - Web Services functionality might be used by other solutions like GMS CallBack, etc. In this case, both Web Services and Web API should be enabled.
-

The General options and descriptions are listed below.

http_port

Location in Configuration Layer by precedence: `web` section of the URS Application

Default value: none

Valid value: any available port

Value changes: take effect on restart

This option is used to enable Web Services functionality. The port is used by HTTP Bridge components to communicate with URS. Missing this option will disable all Web Service functionality.

Note: Alternatively, Web Services functionality can be enabled by specifying the port in the list of ports in the URS application Server Info tab. The port must have the ID "web".

See the *Universal Routing 8.1 Deployment Guide* for information on configuring Universal Routing to work with Workforce Management.

The Log Options and descriptions are listed below.

http_log_file

Location in Configuration Layer by precedence: `web` section of the URS Application

Default value: none

Valid value: log file name

Value changes: take effect on restart.

Log file for HTTP Bridge error and trace messages.

http_log_size

Location in Configuration Layer by precedence: `web` section of the URS Application

Default value: `10000`

Valid value: size of log file in kilobytes

Value changes: take effect on restart

HTTP Bridge maximum log file segment size in kilobytes. Once the specified file size is reached, a new segment/file is created and the new log output goes to this new file. You must also configure the `http_log_file` option to use this option. Also, see the option `log_remove_old_files`.

log_buffering

Location in Configuration Layer by precedence: `web` section of the URS Application

Default value: `false`

Valid value: `true/false`

Value changes: take effect on restart

Set the value of this option to `true` to turn on log buffering.

log_remove_old_files

Location in Configuration Layer by precedence: `web` section of the URS Application

Default value: `20`

Valid value: either `false` (meaning old log files are not deleted and all log files will be kept), or an integer number of log files that will be kept.

Value changes: take effect on restart

This option specifies whether the previous segments/ files are to be deleted when the new segment/file is created.

verbose

Location in Configuration Layer by precedence: `web` section of the URS Application

Default value: `0`

Valid value: `0` to `3`

Value changes: take effect on restart

Level of log output. Level `0` produces no log messages. Levels from `1` to `3` produce log information with a higher level of detail for the higher log levels.

Warning! Warning: Do not confuse this option with other options that have the same name, which are described in Appendix A, “Log Events” on [page 751](#), and “verbose” on [page 684](#).

Warning! Logging functionality is simplified in 8.1. All logging is controlled using `verbose`, `http_log_file`, `http_log_size`, `log_buffering`, and `log_remove_old_files` options. As a result, starting with 8.0, the following options were removed: `xml_log_file`, `xml_log_size`, `parser_log_file`, `parser_log_size`, `wfm_xml_log_file`, `wfm_xml_log_size`, `wfm_parser_log_file`, `wfm_parser_log_size`.

The Security Options and descriptions are listed below.

def_certificate

Location in Configuration Layer by precedence: `web` section of the URS Application
Default value: none
Valid value: certificate
Value changes: take effect on restart
Certificate to use for secure connection if Web Service object does not provide any.

def_certificate_key

Location in Configuration Layer by precedence: `web` section of the URS Application
Default value: none
Valid value: certificate key
Value changes: take effect on restart
Certificate key to use for secure connection if Web Service object does not provide any.

def_trusted_ca

Location in Configuration Layer by precedence: `web` section of the URS Application
Default value: none
Valid value: certificate authority
Value changes: take effect on restart
Certificate authority to use for secure connection if Web Service object does not provide any.

The SOAP options and descriptions are listed below.

soap_conn_idle_timeout

Location in Configuration Layer by precedence: `web` section of the URS Application

Default value: 15 (seconds)

Valid value: time in seconds

Value changes: take effect on restart

Time period HTTP Bridge will maintain persistent SOAP connections without activity, in seconds. When this period expires for a given connection, HTTP bridge closes the connection.

soap_retry_attempts

Location in Configuration Layer by precedence: `web` section of the URS Application

Default value: 2

Valid value: number of attempts

Value changes: take effect on restart

Maximum number of attempts HTTP Bridge makes to communicate with a Web Service.

soap_retry_timeout

Location in Configuration Layer by precedence: `web` section of the URS Application

Default value: 5

Valid value: time in seconds

Value changes: take effect on restart

Time period HTTP Bridge waits between retry attempts, in seconds. The value of `0` (zero) forces HTTP Bridge to initiate the next communication attempt immediately.

The REST options and descriptions are listed below.

http_conn_idle_timeout

Location in Configuration Layer by precedence: `web` section of the URS Application

Default value: 15 (seconds)

Valid value: time in seconds

Value changes: take effect on restart

Time period HTTP Bridge will maintain persistent REST connections without activity, in seconds. When this period expires for a given connection, HTTP Bridge closes the connection.

The WorkForce options and descriptions are listed below.

wfm_polling_interval

Location in Configuration Layer by precedence: web section of the URS Application

Default value: 15

Valid value: time in minutes

Value changes: take effect on restart

Time period in minutes between obtaining agent's schedule data from WFM.

Web API (URS-Behind) Options

To implement the Web API functionality ("URS-Behind" functionality) when URS plays the role of WEB Service and serves REST and SOAP requests from Web Clients, URS uses a component called *HTTP Interface*, which also uses some of the options described in this section.

For more information on the Web API functionality, see Appendix C, “URS-Behind Solution,” on [page 797](#), and the *Universal Routing 8.1 Deployment Guide*.

Note: Web API functionality might be used by other solutions like GMS CallBack, etc. In this case, both Web Services and Web API should be enabled.

The General options and descriptions are listed below.

http_port

Location in Configuration Layer by precedence: http section of the URS Application

Default value: none

Valid value: any available port

Value changes: take effect on restart

This option is used to enable Web API functionality through REST HTTP API. It should be set to an available port, which will be the port to which Web Clients should connect to utilize REST Web API in URS.

Note: Alternatively this functionality can be enabled by specifying a port in the list of ports on the URS application Server Info tab. The port must have the ID `http`.

soap_port

Location in Configuration Layer by precedence: `http` section of the URS Application

Default value: none

Valid value: any available port

Value changes: take effect on restart

This option is used to enable Web API (URS-Behind) functionality through SOAP HTTP API. It should be set to an available port, which will be the port to which Web Clients should connect to utilize SOAP Web API in URS.

Note: Alternatively this functionality can be enabled by specifying a port in the list of ports on the URS application Server Info tab. The port must have the ID `soap`.

The Log Options and descriptions are listed below.

log_file

Location in Configuration Layer by precedence: `http` section of the URS Application

Default value: none

Valid value: log file name

Value changes: take effect on restart

Log file for HTTP Interface error and trace messages, `log_size` HTTP Interface maximum log file segment size in kilobytes. Once the specified file size is reached, a new segment/file is created and the new log output goes to this new file. You must also configure the `http_log_file` option to use this option.

log_buffering

Location in Configuration Layer by precedence: `http` section of the URS Application

Default value: `false`

Valid value: `true/false`

Value changes: take effect on restart.

Set the value of this option to `true` to turn on log buffering.

log_remove_old_files

Location in Configuration Layer by precedence: `http` section of the URS Application

Default value: `20`

Valid value: `false` (meaning old log files are not deleted and all log files will be kept), or an integer number of log files that will be kept.

Value changes: take effect on restart

Specifies whether the previous segments/files are to be deleted when the new segment/file is created.

IRD Options

IRD option descriptions are listed below.

bytecode

Location in Configuration Layer by precedence: IRD

Default value: `false`

Valid value: `true`, `false`

Value changes: take effect upon restart

This IRD option is set in the `default` section of the IRD Application in the Configuration Database. If set to `true`, it causes IRD to read strategy bytecode upon connection to Configuration Server. As a result, when exporting a strategy (see “Exporting Strategies” on [page 46](#)) the bytecode is also exported. Importing the `.zcf` file results in a strategy that is ready to run without recompiling. Setting this option to `false` results in the strategy export function skipping bytecode. In this case, you must recompile the imported strategy in the target environment to make it ready to work.

Note: Because setting this option to `true` can cause IRD to start slowly, Genesys recommends that users working in large environments create a dedicated IRD for exporting/importing strategies and set this option to `true` for that instance of IRD only.

inactivity-timeout

Location in Configuration Layer by precedence: IRD

Default value: `0` (false)

Valid value: `0` or any positive integer reflecting the number of minutes of inactivity, which will cause the IRD Application to require user re-authentication.

Value changes: take effect dynamically

This option allows you to require that users log back into IRD after a specified period of user inactivity as defined in the *Genesys 8.1 Security Deployment Guide*. The default of `0` (zero) makes sense for monitoring applications such as IRD, since you may not wish to stop monitoring interactions Entered, Routed, Abandoned, and In Process (see Figure 9 on [page 39](#)) after a period of inactivity or monitoring strategy object interactions in Trace View (see Figure 10 on [page 40](#)).

To change the default, configure a value for this option. Enter the timeout in minutes.

- IRD interprets the absence of the `inactivity-timeout` option or the setting of its value to `0` (zero) as the disabling of application locking by user inactivity.
- Setting a positive integer enables the security timeout feature in IRD.

When set to a value greater than zero (`0`), users must log back into IRD after the specified period of user inactivity. Inactivity is defined as no keyboard or mouse input when the UI is active.

Note: Starting with the 7.6 release, the `security` section containing the `inactivity-timeout` option is presented by default in the Application Template. If you are migrating from an old template, then you must manually create the `security` section in the IRD Application object. Then define this option in that section.

Important Information

- Upon expiration of the value set for the `inactivity-timeout` option, all opened IRD windows are hidden (minimized). You are presented with a re-login dialog box that, for authentication purposes, asks only for a user password in order to resume the Application session. All other fields are grayed out since the connection of the IRD Application to Configuration Server remains active the entire time until re-login.
If you want to end the Application session by pressing `Cancel` button, on the re-login dialog box you will be warned that you are about to exit the Application. To exit the Application, press the `Yes` button on the `Do you want to close the application?` warning message. To return to the re-login dialog box, press the `No` button.
- If the administrator changes a user's password in Configuration Manager at any time during an IRD session before that user is presented with the re-login dialog box, the previous password will be required in order to resume the suspended session.
- In a case where the user changes a password in IRD before the expiration of the value set for `inactivity-timeout` option, then only a new password will be accepted in the re-login dialog box in order to continue working with the Application.
- The `Interaction Design` window behaves differently than the `Routing Design` window in the case where you want to end the Application session while you have unsaved changes. Unlike `Routing Design`, when you press the `Yes` button in order to close the Application, `Interaction Design` provides a message giving the possibility to save modifications made to a business process prior exiting.

Note: Starting with IRD 8.1.3, support for the ‘Reset Password’ configuration attribute is introduced. It allows the user to be prompted to change the logon password while connecting to Configuration Server. See the Starting Interaction Routing Designer section of the *Universal Routing 8.1 Interaction Routing Designer Help* for more details.

Custom Server Options

Note: For information on deploying Custom Server, see the *Universal Routing 8.1 Deployment Guide*, Chapter 5.

There are two sections in the Custom Server Application object: `default` and `log`. See *Universal Routing 8.1 Deployment Guide*, Chapter 9 for information on the `log` section. The options for the `default` section are described below.

async

Location in Configuration Layer by precedence: Custom Server

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect immediately for new requests only

The `async` option tells Custom Server to call the `R_CUSTOMER_PROCEDURE` of the RCP module in either synchronous or asynchronous mode.

- A value of `true` tells Custom Server to interact in asynchronous mode.
- A value of `false` tells Custom Server to interact in synchronous mode.

The default value is `false`.

buffer_size

Location in Configuration Layer by precedence: Custom Server

Default value: `4096`

Valid values: any non-negative integer

Value changes: take effect immediately

The `buffer_size` option specifies the maximum size of the buffer (in bytes) that contains the result of `R_CUSTOMER_PROCEDURE`. The default (and minimum) value is `4,096` bytes. The actual size of the buffer cannot be reduced. A new size can only be made greater than the old size, but cannot be less than `4K`. Otherwise the value change has no effect and generates a warning.

frequency

Location in Configuration Layer by precedence: Custom Server

Default value: `1000`

Valid values: any non-negative integer

Value changes: take effect immediately

The `frequency` option is valid for asynchronous mode only. This option specifies the time interval in milliseconds that Custom Server waits between two consecutive `R_CUSTOMER_PROCEDURE` calls related to the same incomplete request. The default value is `1000` milliseconds. This option takes effect for all requests.

hide_private_data

Location in Configuration Layer by precedence: Custom Server

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect immediately

Note: URS also provides this option (see [page 643](#)), however it works a little differently when used with Custom Server. This is because Custom Server does not print T-Library events or the results of database access in the log.

If set to `true` this option prevents Custom Server from showing Universal Routing Server request parameters in the log.

life_time

Location in Configuration Layer by precedence: Custom Server

Default value: `600`

Valid values: any non-negative integer

Value changes: take effect immediately

The `life_time` option specifies the time interval, in seconds, during which the requests sent to a custom procedure are assumed to be valid. The default value is `600` seconds. This option takes effect for all requests.

new_parsing

Location in Configuration Layer by precedence: Custom Server

Default value: `true`

Valid values: `true`, `false`

Value changes: take effect immediately

This option provides advanced parsing ability. Use when parameters of requests that Custom Server receives from URS can contain a single quote ('). Its default value is `true` so the advanced parsing is switched on by default. If set to `false`, Custom Server parses data using the previous 5.x/6.x/7.x parsing rules.

Routing-Related Interaction Server Options

When processing e-mails and other non-voice interactions, URS communicates with Interaction Server, a software component that functions as the central interchange for interaction flow in Genesys eServices.

Note: Interaction Server replaces MS T-Server that existed in ICS 6.5.1.

In summary, Interaction Server:

- Receives interaction operational data from the media interface. For example, in the case of e-mail this is E-mail Server Java.
- Stores the operational data in a cache (a database) while receiving and transmitting information about the interaction. This cache also contains queues through which the interaction passes as part of its processing.
- Works in concert with URS to route interactions according to business processes (see [page 149](#)).

Interaction Server options are configured on the `Options` tab of the `Interaction Server Application` object `Properties` window. You can also configure options in the `<Interaction Server>` section of the `Annex` tab for the `Universal Routing Server Application` object, and the `default` and `<Universal Routing Server>` sections of the `Annex` tab for the `Strategy` object (located in the `Scripts` folder in Configuration Manager).

The options in [Table 140](#) affect URS/Interaction Server communication and e-mail processing. For complete information on all Interaction Server options, see the *eServices (Multimedia) 8.1 Reference Manual*.

Table 140: Interaction Server/URS Options

Section	Option
Options Tab	
settings	allow-duplicates-in-submit
	default-max-submission-rate
	default-max-submitted-per-strategy
	default-max-submitted-per-router
	delay-updates
	delivering-timeout
	handling-timeout
	hide-attached-data
	ignore-read-only-on-submit
	low-pull-threshold
	max-interactions-per-pull
	max-interactions-per-snapshot
	max-number-of-snapshots
	not-ready-on-invitation-timeout
	registration-timeout
	routing-timeout
	statistic-interval
	third-party-server-queue-size
	third-party-server-timeout
	third-party-server-window-size
agent-reservation	reject-subsequent-request
	request-collection-time
	reservation-time
log	messagefile

Table 140: Interaction Server/URS Options (Continued)

Section	Option
Annex Tab (Universal Routing Server Application Object)	
<Interaction Server>	max-submitted-interactions
	max-submission-rate
Annex Tab (Strategy Object)	
default	max-submitted-interactions
< <i>Universal Routing Server name</i> >	< <i>Interaction Server Name</i> >.max-submitted-interactions

Routing-Related Message Server Options

URS uses various options set in the Message Server Application object for logging, monitoring, and inter-router communications. See:

- “lds” on [page 645](#)
- “using” on [page 683](#)

Also see “monitoring_time” on [page 646](#).

ADDP Options for Universal Routing Server

Advanced Disconnect Detection Protocol (ADDP) (formerly Keep Alive Protocol (KPL)) is used for detecting a connection failure with servers to which URS connects as a client. This ADDP option must be configured for each server within the Connections tab of the URS Properties dialog box as follows:

1. Launch Configuration Manager.
2. In the Applications folder, double-click the URS Application to open its Properties dialog box.
3. Select the Connections tab.

This tab lists the servers to which URS connects.

For each server do the following:

1. Within the Connections tab of the URS Properties dialog box, double-click the server name to open its properties dialog box.
2. Type addp in the Connection Protocol field.

Protocol specifies the method for detecting connection failures between two or more servers and determining the operation status of these servers. The value addp activates the ADDP.

3. Enter the Local Timeout and Remote Timeout values in their respective fields.

These fields specify the timeout, in seconds, between ceasing to receive an ADDP signal and generating a Server Disconnected event. The specified value must be greater than twice the round-trip ADDP polling time to servers.

Local Timeout is the heartbeat polling interval measured on a client side.
Remote Timeout is the heartbeat polling interval on a server side.

4. From the Trace Mode drop-down list, select a trace level (Trace On Both Sites, Trace On Client Site, or Trace On Server Site).
Trace specifies the level of the ADDP log.
5. Click OK to close the server properties dialog box.
6. After specifying the ADDP option for the servers, click OK to close the URS Properties dialog box.

Note: All ADDP protocol changes (including switch off.on) will take effect only upon connecting/restoring the connection with servers to which URS connects as a client.

Restoring the Connection to Configuration Server

URS can reconnect to Configuration Server in two ways: By using the *from the beginning* mode of connecting and by restoring a previous connection.

Restoring a connection is much faster but not always possible and require a previously successful connection to Configuration Server.

When URS connects to the Configuration Server, the connection is initially flagged *not restorable*. The connection is flagged as *restorable* only after Configuration Server successfully reads all data. The connection remains *restorable* during operation and is available if, for any reason, URS becomes disconnected. The flag can be reset to *not restorable* if URS receives a server error or disconnect message such as, CFGHistoryLogExpired (with a value of 27), from Configuration Server. In this case, the connection cannot be restored and the router switches to *from the beginning* mode of connecting.

Retired Options

The following options are no longer used in Universal Routing:

- close_unused_statistic
- close_statistic_time
- max_loading
- reg_mode

The following options were retired in Genesys 6.x:

- backupserver
- backup_priority_level
- backup_mode (reinstated in 8.1)

Note: The backup options were replaced by the common backup configuration features of Genesys 6.x available for all servers through the Configuration Layer. In 6.x, backup servers were managed by the Management Layer.

- joint_work
- log_file_name
- log_remove_old_files
- log_file_size
- log_buffering

Note: For all log options, refer to the *Framework 8.1 Deployment Guide* (Management Layer Functionality) or to the *Framework 8.1 Solution Control Interface Help*.

- kprl_priority
- kpl-interval
- strategy (retired, but reused in 7.5, see [page 668](#))

Note: The recommended way to give a value to this option is by using the Load Strategy command in Interaction Routing Designer.



Chapter

5

Number Translation

From a Universal Routing standpoint, number translation is the process of converting numbers that are recognized by URS into the numbers that are passed to a switch. If such conversion is necessary such as for external routing, you specify the rules in the **Switch Access Code Properties** dialog box in Configuration Manager.

This chapter includes the following sections:

- [Configuring the use_translation Option, page 705](#)
- [Configuring Translation Tables, page 706](#)

For additional information on this feature, see the chapter on Multi-Site Support in any T-Server deployment guide.

Configuring the use_translation Option

To configure number translation:

- Set the value of the `use_translation` option (see [page 682](#)) to `true`.

Then create a *translation table* for each switch. To do so, open the **Switch Access Code Properties** dialog box for each switch you are using and configure the settings on the **General** tab. These settings are explained in the following sections.

If you set the value of the `use_translation` option to `false`, you can prefix a sequence of digits to a DN before sending it from the switch. To prefix digits:

1. Enter the digits in the `Code` text box on the **Switch Access Code Properties** dialog box (see [Figure 250](#)).
2. Enter the name of the destination switch in the `Switch` field.

Configuring Translation Tables

As described in the chapter on MultiSite Support in the applicable T-Server deployment guide, you can set up a different translation table for each destination switch. A *translation table* consists of all of the values set on the Switch Access Code Properties dialog box's General tab. Figure 250 shows the dialog box for a SIP Switch with the Route Type dropdown menu.

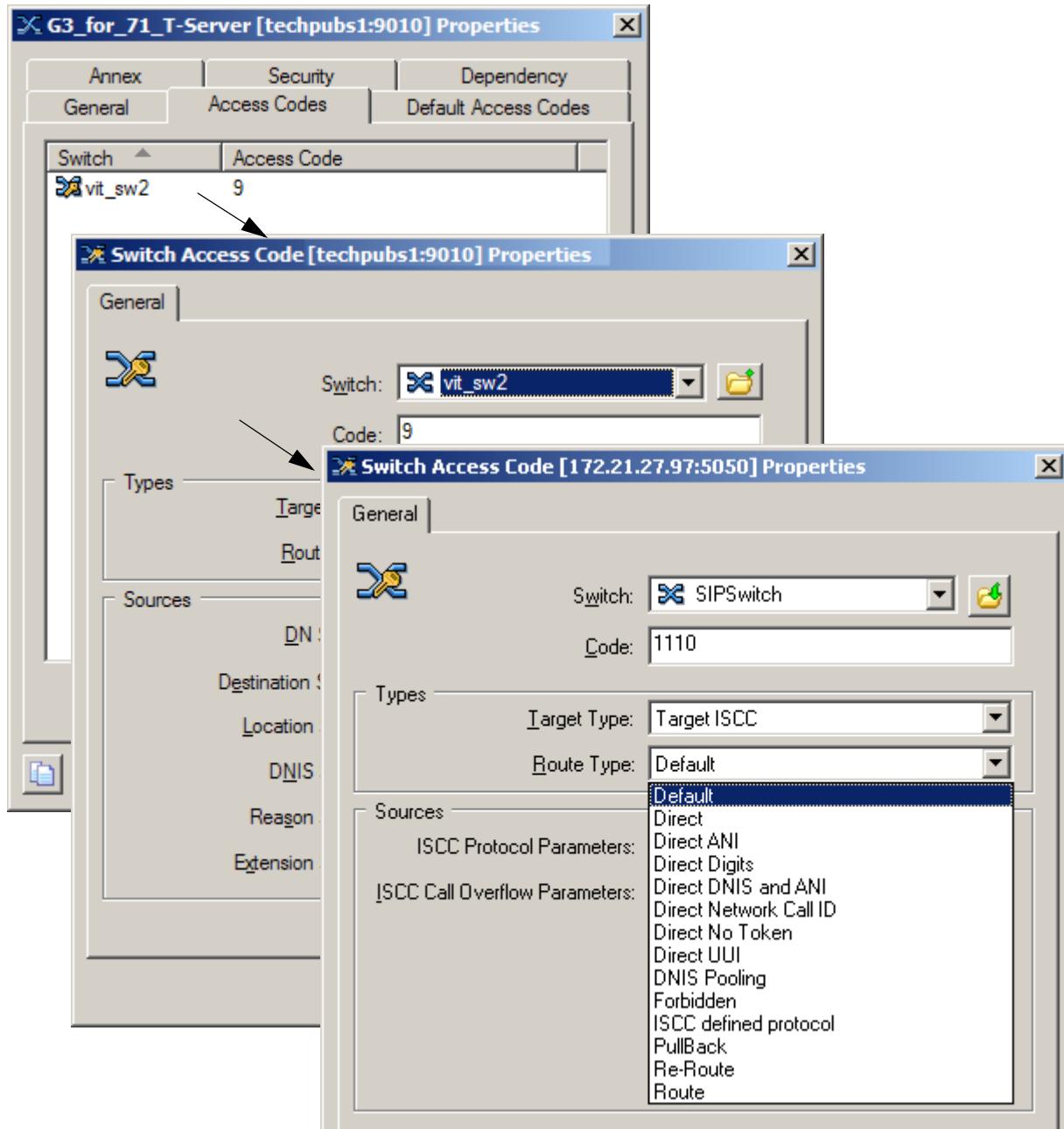


Figure 250: Example Switch Access Code Properties Dialog Box

Selecting a Target Type

The **Target Type** text box adds further versatility. You can set up different translation tables between the same two switches corresponding to the type of targets returned by a strategy.

Selecting a Route Type

Use the **Route Type** drop-down list to select a route type to pass to the destination switch. All the possible choices are explicit route types with the exception of **Get From DN**. Selecting **Get From DN** will extract the Route Type information from the DN Properties set up in Configuration Manager.

The entries in the text boxes that follow must consist of an expression, which may be preceded or followed by fixed characters. The expressions are built of keywords and, possibly, instructions that excerpt relevant characters from the string indicated by the keyword.

Parameters for Switch Access Codes

The **Sources** area in the **Switch Access Code Properties** dialog box shown in Figure 250 on [page 706](#) is where you enter extended parameters for your Switch Access Codes by specifying the **ISCC Protocol** and **ISCC Call Overflow Parameters**.

Source Field Keywords

[Figure 251](#) shows the **Switch Access Code** dialog box for a SIP Switch. Figure shows the dialog box before selecting a switch.

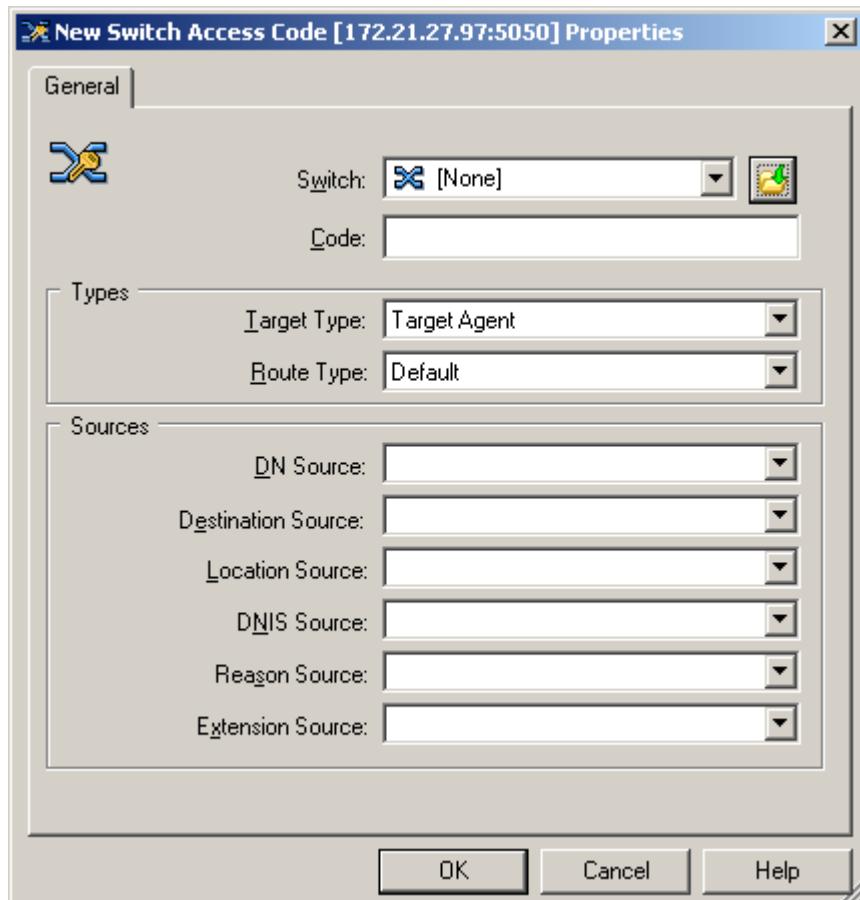


Figure 251: Switch Access Code Dialog Box Before Switch Selection

The possible keywords for the Destination Source field are:

- DN.DL—retrieves a DL
- AGENT.LOGIN—retrieves an agent login number
- AGENT.NAME—retrieves an agent employee ID number
- TARGET.CCTN—retrieves the number associated with the target produced by the CCTExtractTargets function

The possible keywords for the Location Source field are:

- SOURCE.TSERVER
- SOURCE.SWITCH
- DEST.TSERVER
- DEST.SWITCH

The possible keywords for the DNIS Source field are:

- ANI
- DNIS
- CED

- AGENT.DNIS
- DEST.DN.DNIS

The first three possible DNIS keywords recover the respective value from the interaction; the last two recover the value from the Configuration Properties of Agent Logins or DNs. To specify a DNIS in those properties, select the Use Override check box in the Advanced tab of the object's properties dialog box.

The keyword may be followed by an instruction for excerpting specified characters. The instruction can have two formats: (m, n) and (-m, n), where m and n are positive integers. The format (m, n) excerpts all characters starting with character m and ending with character n counted from the beginning of the string, while (-m, n) excerpts characters from m to n counted from the end.

The keyword, along with the excerpt that may be specified after it, must be enclosed in square brackets. For example, entering 123[DEST.DN.DNIS(2,4)] into the DNIS Source box will result in extracting the DNIS from the DN properties, taking its second, third, and fourth digit, concatenating them to 123, and sending the resulting number as a DNIS to the destination switch.

Note: For additional information on completing these ISCC fields, see the chapter on Multi-Site support in the applicable T-Server deployment guide.

Configuring the Reason Source and Extension Source

The Reason Source and Extension Source text boxes should include the T-Server-specific information required by certain switches. Such information is presented as a nested key-value list; in other words, the values of each key can be a list inside a key-value list. The format of the lists as entered in the boxes must consist of items separated by the pipe symbol (|); each item must be of the form

Key1.Key2. . . . KeyN:Value

where Value is not itself a key-value list, and Key1.Key2. . . . KeyN is the sequence of keys in the nested lists that leads to the corresponding value. Key1, Key2, . . . , KeyN, and Value must be of String type.

For information on what information should be entered in these two fields, see the chapter “T-Library Events” in the *Framework 7.5 T-Server Developer’s Guide*.



Chapter

6

Automatically Attached Pegs

Pegs are a special class of key-value pairs that are used as counters for certain actions that occur during the execution of a routing strategy. The key of a peg is the name of the counter, and the value is always an integer.

URS can create two kinds of pegs, both of which are explained in the following sections:

- [Explicit Pegs, page 711](#)
- [Automatic Pegs, page 711](#)

Because pegs reflect the history of the interaction during the execution of the strategy, they are not attached to the interaction until immediately before the interaction is definitively routed. In this way, they are unlike key-value pairs attached by means of the IRD Attach and Update functions, which are always attached to the interaction at the moment when URS executes the corresponding statement. Peg values can be increased an arbitrary number of times before the interaction is definitively routed.

Explicit Pegs

Explicit pegs are created by invoking the URS Peg function (see “Peg” on [page 557](#)). These pegs can have arbitrary names and their values can be increased by any integer that is prescribed as a parameter of the Peg function.

Automatic Pegs

Automatic pegs are attached only if you set the value of the URS `automatic_attach` configuration option to `true` (the default value; see “`automatic_attach`” on [page 632](#)). They have specific names that the user

cannot change, and their values increase after specific actions (except for PegSF, which does not increment).

The values for the pegs that *do* increment are always increased by 1 whenever the action that triggers the increase occurs. [Table 141](#) shows a list of all of the automatically-attached pegs and specifies the circumstances under which their values are increased.

If the action that triggers the increase never happens during the execution of a strategy, the corresponding peg is not attached to the interaction.

Table 141: Automatically Attached Pegs

Name	Increased When...
PegDOW	The Day function is called in the strategy.
PegDOY	The Date function is called in the strategy.
PegTD	The Time function is called in the strategy.
PegQT	The SelectTargets function is called in the strategy.
PegProc<procedure_name>	A database statement that uses a SQL procedure or Custom Server is called in the strategy. Here, <procedure_name> represents the name of the procedure as specified in the strategy.
PegSL<table_name>	A database statement that uses a SQL Select statement is called in the strategy. Here, <table_name> represents the name of the database table from which information is retrieved.
PegDL<directory_number>	<p>The [DN.DL] translation function is called (that is, a network destination is selected as a result of number translation), or the interaction is routed to a target of type DL.</p> <p>The value of this peg is increased at the time of translation; therefore, it can be increased as a result of an explicit call of the URS Translate function.</p> <p>Here, <directory_number> represents the actual network destination that is selected.</p>
PegDEF	The interaction is routed to the default destination.
PegAG<group_name>	A target of type .GA (Group of Agents) is selected. Here, <group_name> represents the name of the group to which the interaction is routed.

Table 141: Automatically Attached Pegs (Continued)

Name	Increased When...
PegLB	The StatLoadBalance statistic is used as a selection criterion in a Routing object or in the SelectDN function. It is also increased when either the SData or PegValue function is called for the StatLoadBalance statistic.
PegPA	Percentage allocation is used as a selection criterion in a Routing object or in the SelectDN function.
PegRejected	The interaction is routed with route type Reject. This peg will also be attached when the URS issues a routing instruction of route type Reject, without executing a strategy. For example, this occurs when no strategy is loaded at a DN that is controlled by URS and the <code>unloaded_cdn</code> option has a value of <code>reject</code> . It can also occur when the <code>validate</code> option has a value of <code>true</code> , and the interaction fails the validation.
PegSF	Unlike other automatic pegs, PegSF does not increment. When a Service Level object is placed in a strategy, URS automatically attaches a new peg to each distributed interaction. This function does not provide real-time information about achieving Service Level specifications. URS calculates the peg value based on an average of the levels for the first 50 calls, or every 30 seconds, whichever comes first. The function is useful for debugging purposes only when determining whether to expand or reduce your group of available agents. You should not use it for reporting about actual Service Level performance at any particular moment. See “Service Level Timers” on page 336 for information on using this peg.



Chapter

7

Routing Statistics

Genesys Stat Server provides URS with information in the form of statistics about the operation of the contact center, including agent availability. URS uses the statistics from Stat Server to route customer interactions. For example, before requesting to route a call to a particular agent, URS checks with Stat Server to verify the agent is available. Also, URS can use statistics, such as the number of calls in queues, to make other routing decisions.

URS uses Message Server to communicate strategy statistics to IRD. This enables users to monitor loaded strategies in IRD.

This chapter includes the following topics:

- [About This Chapter, page 716](#)
- [Predefined Statistics, page 716](#)
- [URS and Stat Server, page 727](#)
- [User-Defined Statistics, page 732](#)

Warning! Deleting or changing the name of any statistic that might be referenced in other strategies, rules, objects, or functions could invalidate the strategy unless the object using the referenced statistic is updated.

Use the Check Integrity tool to validate all strategies and confirm the existence of all referenced statistics. See *Universal Routing 8.1 Interaction Routing Designer Help* for instructions on how to use this tool.

About This Chapter

This chapter introduces the predefined statistics used by various IRD objects, the relationship of URS to Stat Server, and user-defined statistics. However, as many of these statistics are discussed in detail elsewhere, this *Universal Routing 8.1 Reference Manual* refers the reader to the following documents for the most up-to-date statistical information:

- The *Framework 8.1 Stat Server User's Guide* contains the following types of information: statistic configuration options, Actions upon which statistical values are based, Object statuses (DN, Place, Agent, Group), statistical Categories, custom formulas, Campaign statistics, and Virtual Agent Groups.
- The *Reporting Technical Reference Guide for the Genesys 7.2 Release* covers statistics that are used for reporting, including those used in the real-time and historical reporting templates provided by Network Routing and Enterprise Routing.

Predefined Statistics

IRD enables you to create wide range of statistics to be used in routing strategies. However, there is a minimum set of statistics that should always exist in order for URS to function correctly.

IRD creates these required statistics (*predefined statistics*). You cannot modify them.

Note: You use both predefined and user-defined statistics in the same way. The only difference is whether IRD creates them automatically or you create them (*user-defined statistics*)

Table 142, “List of Predefined Statistics,” on [page 719](#), lists the predefined statistics, their attributes, and parameters.

The predefined statistics also appear in the `Statistics` list in Interaction Routing Designer (see [Figure 252](#)).

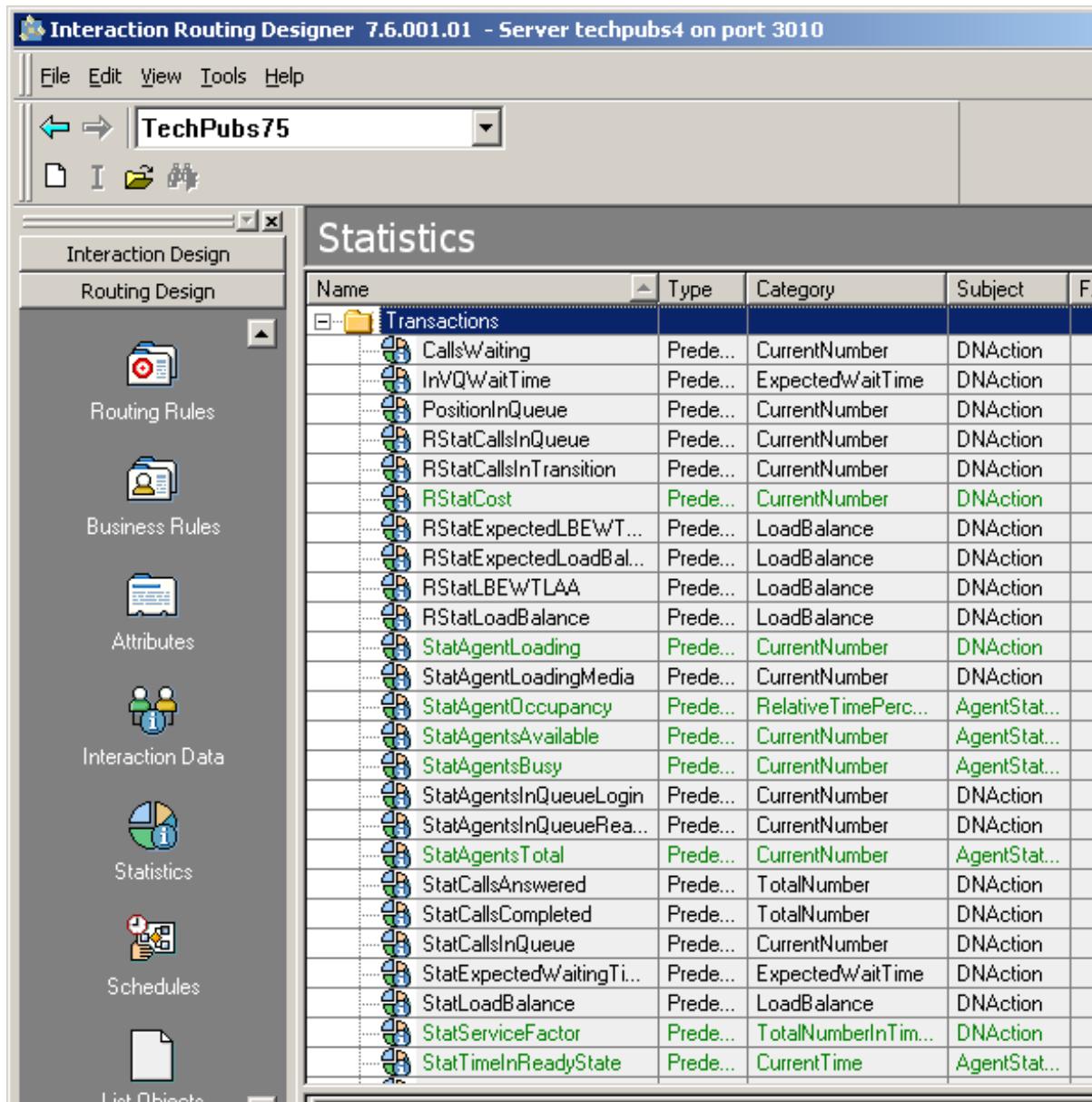


Figure 252: Predefined Statistics in IRD

Use of Predefined Statistics in IRD

The predefined statistics are available for functions that support skill expressions (see [page 88](#)) including CountSkillInGroupEx (see [page 511](#)), GetSkillInGroupEx (see [page 519](#)), MultiSkill (see [page 526](#)), PegValue (see [page 557](#)), ResetStatAdjustment (see [page 569](#)), SData (see [page 572](#)), SelectDN (see [page 608](#)), SetStatAdjustment (see [page 574](#)), and SetThresholdEx (see [page 577](#)).

The statistics shown in [Figure 252](#) also are available for the following IRD objects or functions:

- Statistics object/Statistics routing rule (see [page 346](#)).
- Workforce object/Workforce routing rule (see [page 353](#)).
- Route Interaction (see [page 322](#)), Workbin (see [page 348](#)), and Selection (see [page 326](#)) objects, for Min or Max Statistics and for the Skill target type as part of a skill expression (see [Figure 253](#)).

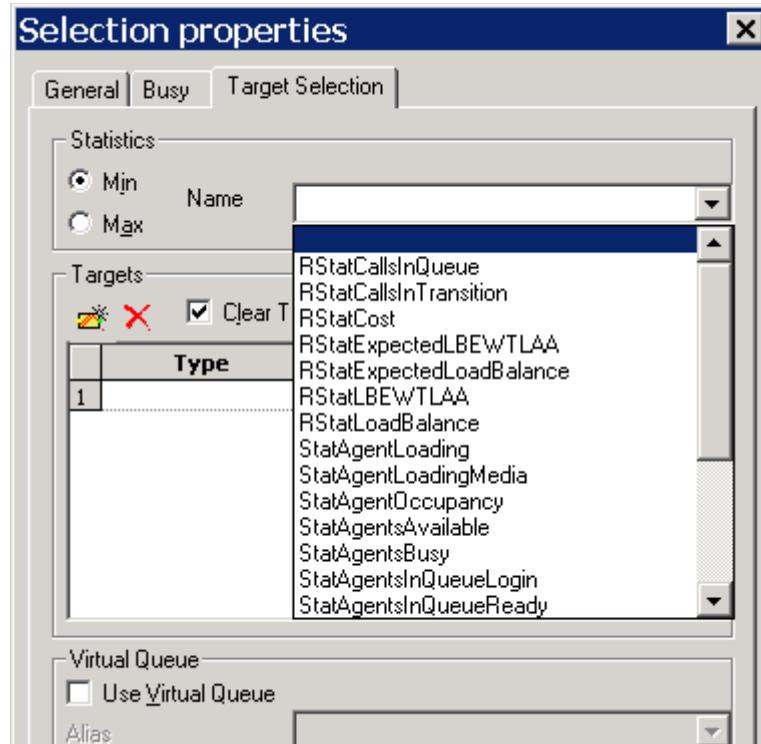


Figure 253: Predefined Statistics in the Selection Object

Note: When routing to a target that is a meta-object (such as a Group of Agents or a Group of Places) and using any Routing object statistic except StatAgentLoading and StatAgentLoadingMedia, URS does not expand the target to its underlying level (for example, the underlying agents belonging to the group). Instead, URS applies the statistic directly to the meta-object.

List of Predefined Statistics

In [Table 142](#), statistics with an attribute of -P are related to Agent, Place, GroupAgents, and GroupPlaces. Statistics with an attribute of -R are related to Queue and Routing Point. When selecting a statistic for the Route Interaction object (see [page 322](#)) or the Selection object (see [page 326](#)), use the appropriate statistic for the selected target type (see [Figure 174 on page 329](#)).

Table 142: List of Predefined Statistics

Statistic	Attribute	Defining Parameters
CallsWaiting (see “Important Notes” on page 721)		For statistics calculated by URS and not Stat Server, Action information is not applicable.
InVQWaitTime (see “Important Notes” on page 721)	-R	For statistics calculated by URS and not Stat Server, Action information is not applicable.
PositionInQueue (see “Important Notes” on page 721)	-R	For statistics calculated by URS and not Stat Server, Action information is not applicable.
RStatLBEWTAA May be used for load balancing (see “Important Notes” on page 721)	-R	For statistics calculated by URS and not Stat Server, Action information is not applicable.
RStatCallsInQueue	-R	For statistics calculated by URS and not Stat Server, Action information is not applicable.
RStatCallsInTransition	-R	For statistics calculated by URS and not Stat Server, Action information is not applicable.
RStatCallsInTransitionEx	-R	For statistics calculated by URS and not Stat Server, Action information is not applicable.
RStatCost (see page 727)	-P	N/A. For statistics calculated by URS and not Stat Server, Action information is not applicable.
RStatExpectedLBEWTAA May be used for load balancing (see “Important Notes” on page 721)	-R	For statistics calculated by URS and not Stat Server, Action information is not applicable.
RStatExpectedLoadBalance (see “Important Notes” on page 721)	-R	For statistics calculated by URS and not Stat Server, Action information is not applicable.
RStatTimeInReadyStateMedia	-P	For statistics calculated by URS and not Stat Server, Action information is not applicable.
RStatLoadBalance	-R	N/A. For statistics calculated by URS and not Stat Server, Action information is not applicable.
RStatRoundRobin	-P	For statistics calculated by URS and not Stat Server, Action information is not applicable.

Table 142: List of Predefined Statistics (Continued)

Statistic	Attribute	Defining Parameters
StatAgentsTotal	-P	Category: CurrentNumber Subject: AgentStatus Action (Main) Mask: All except the following- LoggedOut, all CallObserved masks, allCallForwarded masks
StatCallsAnswered	-P	Category: TotalNumber Interval (Time Profile): GrowingWindow Subject: DNAAction Action (Main) Mask: CallAnswered
StatCallsCompleted	-P	Category: TotalNumber Interval (Time Profile): Growing Window Subject: DNAAction Action (Main) Mask: CallUnknown, CallConsult, CallInternal, CallOutbound, CallInbound
StatCallsInQueue (see “Important Notes” on page 721)	-R	Category: CurrentNumber Subject: DNAAction Action (Main) Mask: CallWait
StatEstimatedWaitingTime (see “Important Notes” on page 721)	-R	Category: CurrentRelativeTimePercentage Interval (Time Profile): GrowingWindow Subject: DNAAction Action (Main) Mask: CallWait
StatExpectedWaitingTime (see “Important Notes” on page 721)	-R	Category: ExpectedWaitTime Interval (Time Profile): Last 600 seconds Subject: DNAAction
StatLoadBalance (see “Important Notes” on page 721)	-R	Category: LoadBalance Interval: Last 50 calls Subject:DNAAction
StatServiceFactor	-R	Category: TotalNumberInTimeRangePercentage Interval (Time Profile): Last 600 seconds Subject: DNAAction Time Range: 0–30 seconds Action (Main) Mask: CallDistributed
StatTimeInReadyState See “Important Notes” below.	-P	Category: CurrentTime Subject: AgentStatus Action (Main) Mask: WaitForNextCall

Table 142: List of Predefined Statistics (Continued)

Statistic	Attribute	Defining Parameters
StatAgentsAvailable	-P	Category: CurrentNumber Subject: AgentStatus Action (Main) Mask: WaitForNextCall
StatAgentsBusy	-P	Category: CurrentNumber Subject: AgentStatus Action (Main) Mask: NotReadyForNextCall
StatAgentsInQueueLogin	-R	Category: CurrentNumber Subject: DNAction Action (Main) Mask: AgentLogin
StatAgentsInQueueReady	-R	Category: CurrentNumber Subject: DNAction Action (Main) Mask: AgentReady
StatAgentLoading	-P	N/A. For statistics calculated by URS and not Stat Server, Action information is not applicable.
• See page 722 for a description.		
StatAgentLoadingMedia	-P	For statistics calculated by URS and not Stat Server, Action information is not applicable.
• See page 724 for a description		
StatAgentOccupancy	-P	Category: RelativeTimePercentage Subject: AgentStatus Action (Main) Mask: CallDialing, CallRing, AfterCallWork
• See page 723 for a description.		
RStatAgentsReadyMedia	-P	For statistics calculated by URS and not Stat Server, Action information is not applicable.
RStatAgentsReadyvoice	-P	For statistics calculated by URS and not Stat Server, Action information is not applicable.

Important Notes

- The statistic, `StatEstimatedWaitingTime`, used in versions 5.1.x and 6.0, has been replaced by `StatLoadBalance` in the `LoadBalance` statistic category. For information on `StatLoadBalance`, see the chapter on load balancing in the *Universal Routing 8.1 Deployment Guide*.
- Stat Server provides a `StatExpectedWaitTime` statistic from the bottom of the queue (in other words, the interaction that just entered the queue). When URS needs a statistic from the top of the queue (the interaction it might be about to route), it divides `StatExpectedWaitTime` by the number of calls in queue to get the predicted wait time for the interaction at the top of the queue.

- If configuring Skills- or Agent-level routing at a site that has a large number of agents, Genesys recommends using the `StatAgentLoading` statistic to select agents. Using the `StatTimeInReadyState` statistic can result in high CPU usage even when there are no calls.
- It is the responsibility of the user to select the appropriate statistic. If the target is an agent group, then use statistics meaningful for groups. If targets are agents or skill expressions, then use statistics meaningful for agents. If targets are queues, then use statistics meaningful for queues. If you do not select a meaningful statistic for the target type, you may receive an error message from Stat Server and the statistic is not opened (or the statistic is opened, but does not make sense for the target).
- The `CallsWaiting`, `InVQWaitTime`, and `PositionInQueue` statistics are used as placeholders in the `SetStatUpdate` function for values returned by the `CallsWaiting`, `InVQWaitTime`, `PositionInQueue` functions, respectively.
- **Warning!** A Cost-Based Routing solution or a Share Agent by Service Level Agreement solution, as described in the *Universal Routing 7.6 Cost-Based Routing Configuration Guide*, is incompatible with load balancing based on `RStatExpectedLoadBalance`, `RStatLBEWTLAA`, `RStatExpectedLBEWTLAA`, `StatExpectedWaitingTime`, `StatEstimatedWaitingTime`, `StatLoadBalance`, `StatCallsInQueue`, other statistic derived from these statistics, or any type of statistic that leads to equal or quantifiable distribution of interactions to routing targets.
- URS and Outbound Contact Server (OCS) can cooperate for the purpose of sharing multi-skilled Agents and/or Places. A special statistic, `CurrentAgentAssignment`, provides information about the activity that the Agent/Place is currently busy with (assigned to). For more information, see the option “environment” on [page 639](#). Also see “Campaign Group Targets” on [page 362](#).
- For information on `RStatCallsInQueue`, `RStatLBEWTLAA`, `RStatExpectedLBEWTLAA`, `RStatExpectedLoadBalance`, `RStatLoadBalance`, and `StatLoadBalance`, see the chapter on load balancing in the *Universal Routing 8.1 Deployment Guide*.

StatAgentLoading

Note: This statistic works for Agents and Places only; it is not applicable for Agent Groups. `StatAgentLoading` always uses a random target order, even if you set the `targets_order` option to `true` (see [page 668](#)).

The `StatAgentLoading` statistic is used to select agents within a group based on current agent loading, which is calculated as a vector of three values: number of busy DNs, time in Ready state (the same value as `StatTimeInReadyState` provides), and a random number.

How It Works

Assume you have multiple agents available in a group and URS must select one. URS always selects an agent in the group according to `StatAgentLoading`. It does this as follows:

- First URS evaluates number of busy DNs and selects the agent with the lowest number of busy DNs.
- If the number of busy DNs is equal among the agents, URS evaluates the second value (time in ready state) and selects the agent with the longest time in ready state.
- If time in ready state is equal among the agents, URS uses a random number.

`StatAgentLoading` is also valuable in an eServices environment, where an agent can handle multiple interactions of different media types. It ensures that URS will always distribute interactions evenly among agents. See [page 724](#) for a comparison of `StatAgentLoading` and `StatAgentLoadingMedia`.

For information on calculating the minimum or maximum value of this statistic when selected in the Statistics, Route Interaction, or Selection Routing object, see [page 326](#).

StatAgentOccupancy

Note: You must be running Stat Server 7.0 or higher to use this statistic.

The `StatAgentOccupancy` statistic enables URS to route interactions to the least occupied agent, which is the agent with the lowest *occupancy rate*.

- *Occupancy rate* is the ratio between the time the agent has been busy since last login relative to the agent's total login time.

The `StatAgentOccupancy` statistic enables URS to evaluate multiple available agents and select the least occupied agent so that the workload among available agents is balanced.

Note: `StatAgentOccupancy` is defined with the `SinceLogin` interval and can be used only in statistics being opened against an Agent object. You cannot use `StatAgentOccupancy` with the Agent Group object type.

Use Case

- After login, Agent #1 was on a call for 5 minutes and was in a Ready state for 5 minutes. His occupancy is 50% ($5/(5+5)$).
- After login, Agent #2 was on a call for 1 minute and was in a Ready state for 2 minutes. His occupancy is 33%($1/(1+2)$).

- An incoming interaction will be distributed to Agent #2 as the least occupied agent.

StatAgentLoadingMedia

Note: The StatAgentLoadingMedia statistic, the UseMediaType function (see [page 479](#)), the GetMediaTypeName function (see [page 474](#)), and the use_agent_capacity option ([page 674](#)) all support the Genesys agent capacity interaction distribution model. To use this statistic, you must be running Stat Server 7.0.1 (which introduces agent capacity rules) or higher.

The StatAgentLoadingMedia statistic works like “[StatAgentLoading](#)” with one difference: Only DNs having the current call media type are counted (StatAgentLoading counts all busy DNs).

For example, if the agent capacity vector indicates that an agent is currently working with one voice interaction and three e-mail interactions then StatAgentLoading returns 4 as first component. StatAgentLoadingMedia returns 1 if asked from a strategy that handles voice interactions.

For information on calculating the minimum or maximum value of this statistic when selected in the Statistics, Route Interaction, or Selection Routing object, see [page 326](#).

RStatAgentsReadyMedia

The RStatAgentsReadyMedia statistic is applied to the current interaction’s media, and counts the number of ready agents according to their configured capacities. The agent is considered ready if its capacity vector for a specific media has a positive margin (number of calls that could be routed to the agent).

This statistic can be applied to agent, places, groups of agents and places, and to skill expressions (groups created by function CreateSkillGroup).

RStatAgentsReadyvoice

The RStatAgentsReadyvoice statistic is applied to voice media, and counts the number of ready agents according to their configured capacities. The agent is considered ready if its capacity vector for a specific media has a positive margin (number of calls that could be routed to the agent).

This statistic can be applied to agent, places, groups of agents and places, and to skill expressions (groups created by function CreateSkillGroup).

RStatRoundRobin

This statistic allows quick selection of a target from the group of agents by applying a round-robin algorithm that provides equal target selection.

URS searches through the list of agents in the target selection object and returns the first available agent. For the next call, URS continues from that selected agent and goes on to the next available agent.

When URS reaches the end of the list of agents, it continues from the beginning of the list. The target selection process continues in a cycle, looking for the first available agent for each call, from the point at which it stopped for the previous call.

RStatTimeInReadyStateMedia

For agent or place, this statistic returns the time in seconds which have passed, beginning when the agent/place time was last reported as able to accept at least one interaction of the current media.

For agent group/place group, it returns the maximum value from the values which are applied to separate agents/places.

Load Balance Family of Statistics

Load balancing between routing targets is essentially *equal* distribution of interactions among ACD queues. *Equal* can refer to the number of calls, the percentage of busy agents, the expected waiting time, or something else. There are many different criteria. What is equal for one customer will not be equal for another. The family of load balance statistics described below use some of more common criteria, but cannot cover all criteria. If the supplied statistics are not appropriate, then you must configure a strategy to implement the criteria you prefer.

[Table 143](#) lists IRD predefined statistics for use in strategies that can be used for load balancing including those that address calls in transition.

Table 143: Load Balancing Statistics

Predefined Statistic	For 7.6 or later?
RStatCallsInQueue	No
StatLoadBalance	No
RStatLoadBalance	No
RStatExpectedLoadBalance	Yes

Table 143: Load Balancing Statistics

Predefined Statistic	For 7.6 or later?
RStatLBEWTLAA	Yes
RStatExpectedLBEWTLAA	Yes

For information on these statistics, see the Load Balancing Chapter in the *Universal Routing 8.1 Deployment Guide*.

Router Self-Awareness

In general, calls are not expected to arrive at their destination immediately after a routing request, but after some (usually short) period of time has elapsed. As result, in any given moment there is a certain amount of *calls in transition*.

Calls in transition (if neglected) can sometimes cause a discrepancy in statistics used by URS for decision making. This, in turn, can affect load balancing or routing. The potential for this situation is amplified when there are multiple concurrently deployed URS instances in load-sharing mode.

Previous to 7.6, the number of calls in transition could apply only to calls routed by the current URS (by the URS that calculates this statistic). Starting with 7.6, you can configure URSs, such as those in a load balancing scenario, to share routing information between themselves by setting a mode called *Router Self-Awareness* as described in the *Universal Routing 8.1 Deployment Guide*. If Router Self-Awareness is activated for the participating URSs, then the number of calls in transition sent to this target will include calls in transition sent by all URSs participating in the same Router Self-Awareness group.

RStatCallsInTransition

Starting with 7.6, URS provides a statistic that cannot be used for load balancing distribution, but can be used in a strategy to adjust other statistics received from Stat Server (similar to RStatCallsInQueue). The RStatCallsInTransition statistic:

- Works for “queue like” targets (ACDQueue, Routing Point) as well as for Agent, Place, and Agent/Place Group.
- Returns the number of calls that URS believes are on the way to corresponding targets, but not yet arrived.
- Also includes calls distributed by all participating URSs when Router Self-Awareness (see [page 726](#)) is activated for this statistic.
- Is counted by URS based on available information from T-Servers.

- Every time URS selects a call to be routed to a target, the corresponding counter for this target is incremented by 1.
- Every time URS detects that a call has arrived at the target, the counter is decremented by 1.
- When URS deletes the call from its memory, the counter is also decremented.

RStatCallsInTransitionEx

Since the RStatCallsInTransition statistic does not take into account calls with issued agent reservation requests, this statistic is not always suitable for use as part of target threshold expressions. To overcome this limitation, the built-in statistic RStatCallsInTransitionEx is provided starting from release 8.0.0 of Universal Routing Server. This statistic behaves like the existing RStatCallsInTransition statistic, but also counts those calls which—while not yet sent to the target—already have an active reservation request issued to this target or to one of the target's members (if the target is a group of agents/places).

RStatCost

You can activate a Cost-based Routing solution by selecting the predefined IRD statistic RStatCost in the Target Selection tab of the Routing Selection object. For more information, see the chapter on cost-based routing in the *Universal Routing 7.6 Cost-Based Routing Configuration Guide*.

URS and Stat Server

Note: Stat Server currently supports virtual group functionality with two types of agent parameters: a Skill configured for an agent and an ACD Queue to which an agent is logged in. You can simultaneously specify both types of parameters in an expression for a single virtual group. For more information, see the Virtual Agent Groups chapter in the *Framework 8.1 Stat Server User's Guide*.

URS uses statistics requested from Stat Server a number of different ways in a routing solution including:

- to determine a target's availability for receiving an interaction and if more than one target is available, which target to route an interaction to.
- to determine which agents to consider for routing for skill-based routing.
- if interactions are in queue (internal as opposed to an ACDQueue), for determining which interaction to route first based on priority.

- when using a Service Level object in a strategy, to determine whether to expand or reduce an agent group to maintain the Service Level requirement.

Stat Server and Statistics

URS's ability to route interactions is affected by its ability to get statistic information from Stat Server. Two statistic issues that can affect routing include:

- Invalid statistics
- Persistent statistics

Invalid Statistics

Stat Server returns an error or an invalid statistic if URS tries to open a statistic for an object that Stat Server is unable to obtain information about. Two reasons might be:

- Stat Server is unable to register the DN or all DNs associated with a Statistical object (such as Place Group) with T-Server.
- Stat Server does not have a connection with the T-Server to which the DNs belong.

If a strategy includes a target of a place group or agent DN that Stat Server has not registered, Stat Server cannot provide URS any information about that target to route the interaction. So, when URS requests information, Stat Server returns an error. The only way to resolve this is to make sure that all underlying DNs used in strategies are registered to Stat Server and have a current connection to the T-Servers involved.

Persistent Statistics

Stat Server treats all requested statistics as *persistent statistics*. Persistent statistics are those that when initially requested are calculated continuously even after URS closes the statistic. As a result, when URS requests the statistic again, Stat Server sends a current value for the statistic. Every time a new statistic is requested it is added to the Persistent Statistic Pool. By default, statistics remain in this pool for three days after the last time it was requested by URS. After this period the statistic becomes obsolete and is removed from the pool.

Note: `OldStatsRemoveInterval` is the Stat Server option that specifies when a statistic becomes obsolete and is removed from the pool. See the *Framework 8.1 Stat Server's User's Guide* for information on setting this option.

Two issues with respect to persistent statistics include:

- A removal of a statistic from the Persistent Statistic Pool could result in an interaction being default routed if the value for the statistic is not available. For example, an agent logs in and `AvgTalkTime` statistic is requested for use in routing. This statistic is added to the Persistent Statistic Pool and calculated continuously. If the agent logs out, Stat Server stores the most recent value for the statistic and when the agent logs back in, that stored value is used as the starting value for `AvgTalkTime`. But if the agent logs out for a long time (longer than 3 days if the default value is used), Stat Server does not continue storing this value. Without this value, the interaction may be default routed.

Other Routing Issues

The following issues can also cause routing problems:

- Object state never changes
- Default routed calls
- Interactions routed to busy DNs
- Interactions not routed when agents are available

Object State Never Changes

If an object remains in a certain state, it typically means that Stat Server did not receive an event that required it to clear the state.

Examples of these include:

- An ACD Queue, with the particular ConnectionID or PreviousConnectionID (in the case of a transfer that is completed to the ACD Queue) that received an `EventQueued` message but never later received an `EventAbandoned` or an `EventDiverted` message.
- A routing point, with the particular ConnectionID or PreviousConnectionID (in the case of a transfer that is completed to the routing point) that received an `EventRouteRequest` but never later received an `EventAbandoned` message or an `EventRouteUsed` message.
- A normal DN that received an `EventOffHook` message but never later received an `EventOnHook` message.

The failure of the correct events to reach Stat Server may be due to interaction flows that are not supported by Stat Server. Interaction flows should be evaluated to determine where the problem lies.

Possible causes include:

- Interactions are routed through DNs that Stat Server does not monitor so some events are never received from the switch.

- Failed DN registrations made by Stat Server to T-Server result in missing events for these DNs. In this case, you must examine why the DN registrations failed.
- Missed switch messages, which indicate a possible problem with switch configuration.
- The loss of link connection between the T-Server and the switch.
- The loss of the connection to T-Server, especially for Virtual Queues where URS generates the respective events for Virtual Queues.

Some serious impacts on URS include:

- If an agent gets into a state where Stat Server thinks the agent is not available, URS will never route an interaction to that agent.
- If Stat Server thinks that there is always an interaction in a queue because it missed some events and the strategy is designed to look at the number of interactions in queue or the age of the oldest interaction queued across a number of targets to determine where to route the interaction, URS may stop sending interactions to a destination.
- If Stat Server always thinks there is an agent available at a location, then all interactions will be sent there, flooding the site, while not sending interactions to other places.

Default Routed Calls

The most common reason that interactions are default routed is due to the [Timeout](#) (waiting time) value being set to [zero](#) (the default) in the Selection object (see Figure 174 on [page 329](#)). When this happens, URS asks Stat Server for a value to route across targets but there is not enough time for the value to be received, so interactions are regularly default routed. Transition time and verification time also may affect whether an interaction is default routed. Also see “[Default](#)” on [page 316](#) and the configuration option `default_destination` on [page 638](#).

Note: In certain circumstances, URS might freeze briefly (for about 2 seconds) when it receives a notification from Configuration Server about a new Data Access Point. During the period when URS is frozen, interactions are default routed.

Interactions Routed to Busy DNs

One reason interactions might be routed to busy DNs is due to competing delivery methods. For example, both URS and an ACD may try to route interactions to the same agent. The agent has logged into a queue to go ready/not ready to show availability to Stat Server for routing. However, this queue is also used to deliver interactions. So both ACD and URS are competing with ACD usually winning. This means that when the agent

becomes ready, the ACD sends a call to the agent. If URS also attempts to route another interaction to the same agent, URS will not succeed because the agent has already received the ACD-delivered interaction.

Interactions Not Routed when Agents are Available

Interactions are not routed to an available agent if Stat Server sees the agent as unavailable (for example, logged out, in a CallRinging state, or some other busy state). Two possible causes for this include:

- The link to the switch goes down and the agent does not log out and log back in. Without this step, Stat Server never sees the agent as available and URS never routes an interaction to that agent.
- The agent is incorrectly in a state that is not available. Look at the Stat Server log for the last status for this agent and then work backwards to determine how the agent got into this state.

Resolving Routing Issues

In order to resolve routing difficulties, you should have available the URS log, Stat Server log, T-Server log, DB Server log, and the strategy. In the URS log, do the following:

- Look for interactions in which problems occurred
- Using the ConnID of a misrouted interaction, examine each line of the log.
- Look for any abnormal entries such as:
 - Targets being ignored or blocked
 - Treatments that are not applied
 - Statistics that appear incorrect
 - Failure in the database lookup
 - Pegs that appear incorrect
 - DNs not functioning (Stat Server log)
 - Impact of URS options `transition_time` and `verification_time` causing routing issues
 - DN types that are incorrect for the targets
 - Default routed interactions
 - Server connection problems

Failure to Open a Statistic

If URS fails to open a statistic (not fails to create one) it provides the following logged message:

```
· statistic <Statistic Name> opening error: reason where reason can be  
one of following: "wrong or missed statdefintion" or "closed server".
```

Failure to open a statistic can occur for two reasons:

1. Wrong or missed `statdefinition`—URS did not find a statistic definition object (normally created by IRD) in the configuration data or the object is found, but its content is corrupted.
2. Closed server—URS is not connected to the Stat Server, on which the statistic must be opened. Stat Server is either not operational or not in the URS connection list.

User-Defined Statistics

The Statistics list in Interaction Routing Designer (see Figure 252 on [page 717](#)) provides many possible combinations for creating statistics. This chapter contains a list of suggested statistics that might be useful in routing strategies, along with information that must be entered into the `Statistics` dialog box.

- For more information about creating statistics in IRD, see *Universal Routing 8.1 Interaction Routing Designer Help*.
- Contact Genesys Customer Care if you need to create a new statistic, but cannot determine what values to use when configuring it in IRD.

Note: Changes to user-defined statistics are not activated until after a restart of the Universal Routing Server.

Creating a New Statistic

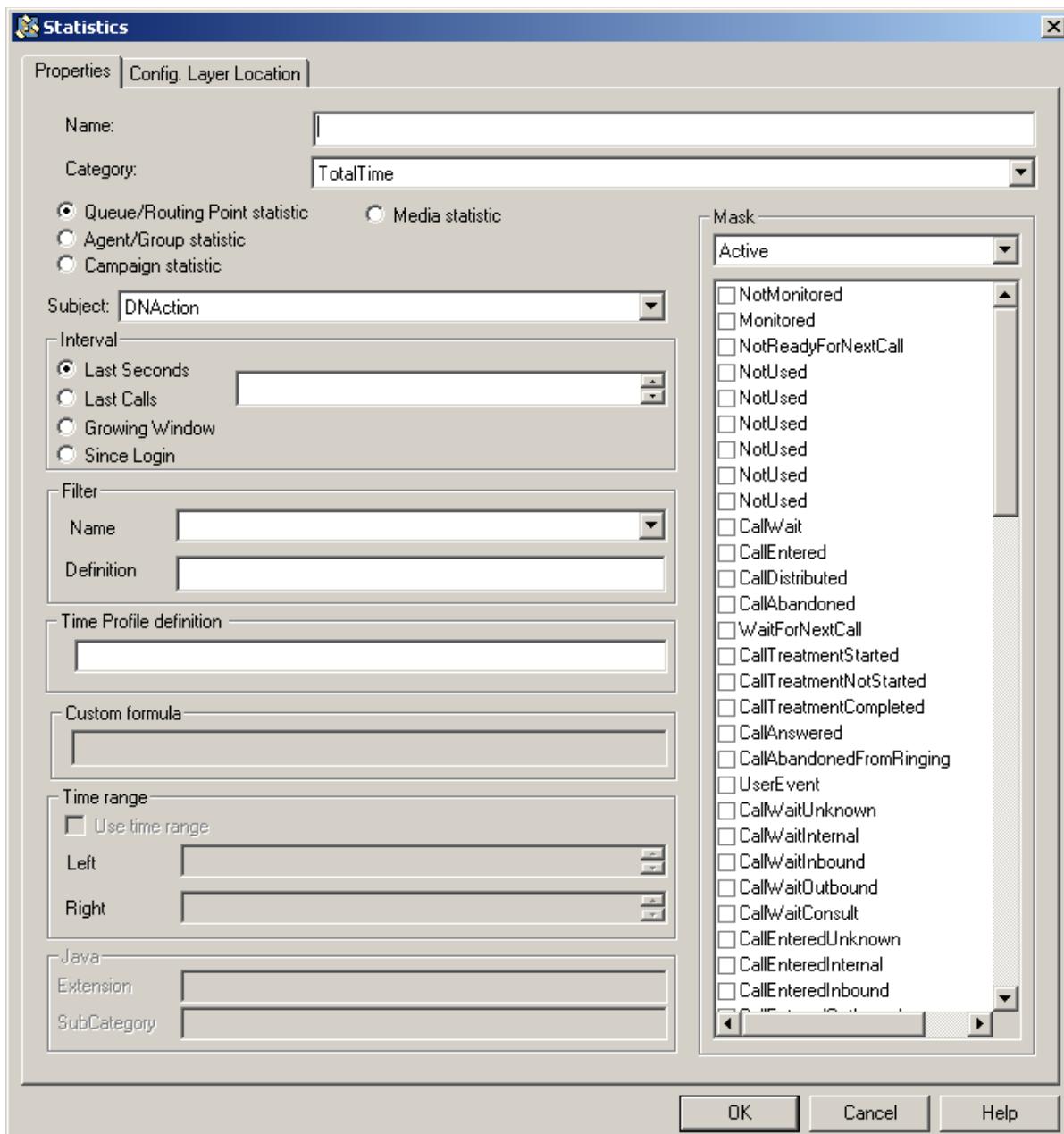
Note: IRD's dialog box for defining custom statistics allows defining statistics based on Java Extensions. You can now enter parameters for `JavaCategory`.

Once you create a new statistic, it becomes available for use in the properties dialog box for the Routing objects that use statistics: “Route Interaction” on [page 322](#), “Selection” on [page 326](#), “Service Level” on [page 333](#), “Statistics” on [page 346](#), and “Workbin” on [page 348](#).

User-created statistics also appear in the IRD main window when you click the Statistics icon (see Figure 252 on [page 717](#)).

To create a new statistic:

1. On the `Routing Design` shortcut bar at the left side of the IRD main window (see Figure 5 on [page 35](#)), click the `Statistics` icon.
2. Select `File > New` from the menu. The `Statistics` dialog box opens (see [Figure 254](#)).

**Figure 254: Statistics Properties Dialog Box**

Select the appropriate radio button to configure one of the following types of statistics:

- Queue/Routing Point
- Agent/Group
- Campaign

Statistics Properties Dialog Box

Fields in the **Statistics** dialog box are summarized below.

Name	This field enables you to name the new statistic.
Category	The statistical category tells Stat Server how to calculate a statistic. For information on statistical categories, see the “Statistical Categories” chapter of the <i>Framework 8.1 Stat Server User’s Guide</i> . In the IRD Statistics dialog box, you can select one of the following categories:
	<ul style="list-style-type: none"> • AverageCustomValue • AverageNumberPerRelativeHour • AverageOfCurrentNumber • AverageOfCurrentTime • AverageTime • CurrentAverageCustomValue • CurrentAverageTime • CurrentContinuousTime • CurrentCustomValue • CurrentDistinctNumber • CurrentMaxCustomvalue • CurrentMaxTime • CurrentMinCustomValue • CurrentMinTime • CurrentNumber • CurrentNumberInTimeRange • CurrentNumberInTimeRangePercentage • CurrentRelativeNumberPercentage • CurrentRelativeTimePercentage • CurrentState • CurrentStateReasons • CurrentTargetState • CurrentTime • ElapsedTimePercentage • EstimTimeToComplete • EstimTimeToEndCurrentNumber • ExpectedWaitTime • Formula • JavaCategory • LoadBalance • LoadBlance1 • MaxCustomValue • MaxNumber • MaxTime • MinCustomValue • MinNumber • MinTime • RelativeNumberPercentage • RelativeTimePercentage • RelativeNumber • ServiceFactor1 • TotalAdjustedNumber • TotalAdjustedTime • TotalCustomValue • TotalNumber • TotalNumberErrors • TotalNumberInTimeRange • TotalNumberInTimeRangePercentage • TotalNumberPerSecond • TotalTime • TotalTimeInTimeRange

Mask

A *mask* is a list of actions or statuses that apply to the statistic category. Depending on the radio button you select (Queue/Routing Point, Agent/Group, or Campaign), the Mask pane lists different actions or statuses.

Where several masks are listed in the description of a statistic, select all of them in the Statistics Properties dialog box. Statistics for which the relative mask column is empty in do not require you to select a relative mask in the Statistics Properties dialog box.

- For lists of actions and statuses, see Chapter 4, “Stat Server Actions” and Chapter 5, “Object Statuses” in the *Framework 8.1 Stat Server User’s Guide*.

Media Statistic

If you select this check box, only the subjects and masks appropriate for media statistics appear in the Subject, Active Mask, and Relative Mask fields. In that case, values for the statistic are collected only for statistics facilitated by MCR Extension components.

Subject

Specifies the statistical objects types that, when changed, affect the statistic. You must specify a value for this option.

Note: The AgentStatus and PlaceStatus objects were the same in releases 5.1, 6.0, and 6.1. However, they are independent in 6.5 and all later releases.

Default Value: No default value.

Valid Values: DNAction, DNStatus, AgentStatus, GroupStatus, PlaceStatus.

Changes Take Effect: When Stat Server is restarted.

For more information, see the Subject of Calculation section in the “Statistical Categories” chapter of the *Framework 8.1 Stat Server User’s Guide*.

Interval

Interval refers to the time interval used for calculating historical aggregate values for statistics. Clients, such as CCPulse+, specify which defined time profile to use when requesting their statistics. Options are Last Seconds, Last Calls, Growing Window, and Since Login.

For more information on intervals, see the Time Profiles section in the “Statistic Configuration Options” chapter of the *Framework 8.1 Stat Server User’s Guide*.

Note: In most routing decisions, the interval over which statistics are measured (if applicable) should have a fixed length. The Growing Window option is intended for exceptional cases and should be used infrequently.

Filter

Available filters appear for selection. Filters enable you to exclude interactions based on certain criteria specified in a logical condition. They enable you to restrict the Stat Server actions that are taken into account during the computation of aggregate values. In a filtered statistic, only those actions are considered that satisfy a filter condition on certain attributes of T-Events, such as DNIS, ANI, CustomerID (or TenantID), MediaType, ThisQueue, TreatmentType, UserData, Reasons, and ExtensionReasonCode.

A filter definition may be provided instead of providing the name from a preconfigured file as well as a definition for the time profile instead of only the default time profile. For Custom categories of statistics (the ones that have word Custom in their name) it is now possible to provide a formula to calculate statistic values based on call attached data.

For more information, see Filters Section in the “Statistic Configuration Options” chapter of the *Framework 8.1 Stat Server User’s Guide*.

Time Profile Definition

Allows to customize time profiles.

Custom Formula

Allows a formula to be provided to calculate statistic values based on a call’s attached data.

Time Range

Stat Server defines the time ranges for collecting data used for several statistical types. In this dialog box, it is possible in URS/IRD 8.0.1 and later *not* to specify a time range at all. If the `Use time range` check box is cleared (not selected), time range will not be set. To use a time range for your custom statistic, make sure to select the `Use time range` check box, then set your time range values.

For more information, see the Time Ranges Section in the “Statistic Configuration Options” chapter of the *Framework 8.1 Stat Server User’s Guide*.

Pseudo Statistics

URS accepts as target selection criteria not only statistics, but alternative criteria, such as agent skills and object properties (“pseudo statistics”). For example, it is possible to instruct URS to select an agent having the maximum skill.

Initiate the pseudo statistic functionality as follows:

1. Create the variable that will contain the pseudo statistic value. Click the X= icon in the Routing Design window and define the variable that will display as a statistic name in the Target Selection tab of the Routing Selection object. [Figure 255](#) shows an example Variable list dialog box.

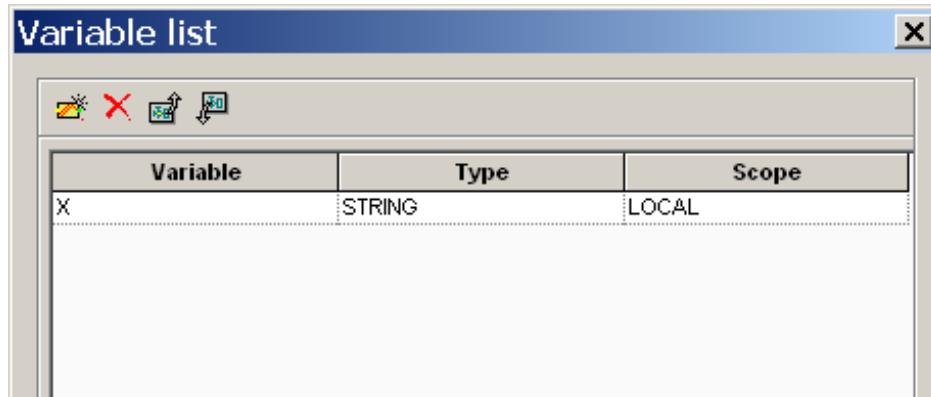


Figure 255: Variable List Dialog Box

2. Use the Assign, Multi-Assign, or Function object to assign a value to the variable using an expression in round brackets surrounded by single quotes. URS interprets a statistic with a name in round brackets as a pseudo statistic. The expression can be:

- the name of a Skill,
- cfgdata (folder, property, default value)

Examples:

- x= '(Spanish)' where the value of Skill Spanish will be used as selection criteria (if target is not agent or has no such skill then 0 will be used)
- x= '(cfgdata(Cost_Folder, RoutingCost, 7.25))' where the value of property RoutingCost of target in folder Cost_Folder will be used as selection criteria.

For example:

```
x='(Spanish)'
```

[Figure 256](#) shows the value assigned to a variable in the Assign properties dialog box.

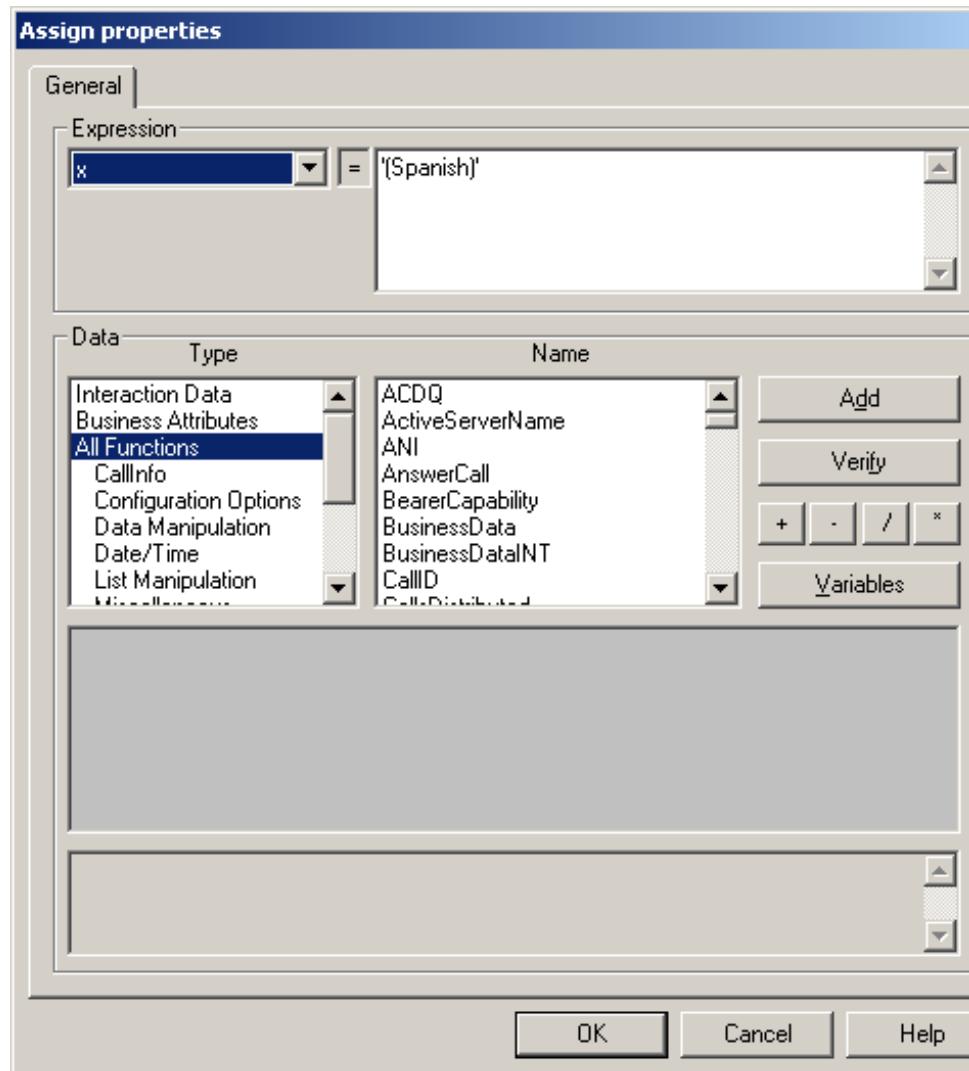


Figure 256: Assign Object, Assigning Skill to a Variable

3. Place and open the Routing Selection object, click the Target Selection tab, select Min or Max, then click the down arrow to list available statistics (both regular and pseudo statistics). [Figure 257](#) shows an example.

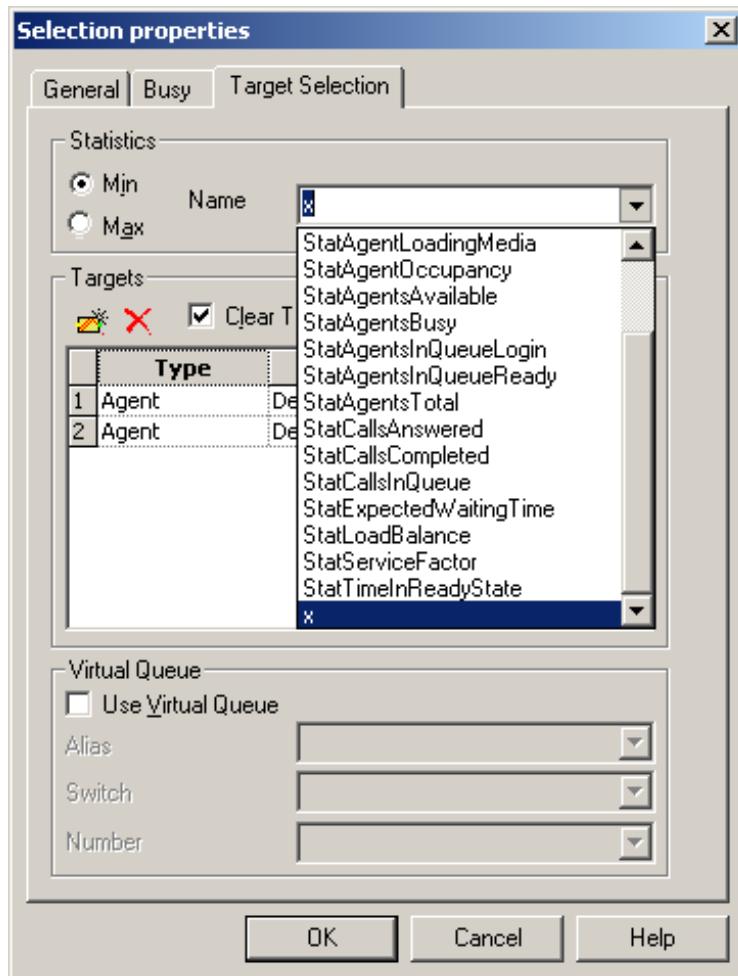


Figure 257: Routing Selection Object, Selecting Pseudo Variable

In this example, URS will use the maximum (Max) value (Skill Level) of the Skill contained in variable *x* as additional selection criteria during target selection. If the target is not an agent or has no such skill, then URS uses 0.

Note: When using pseudo statistics, it is the user's responsibility to determine the reasonability of combining a certain type of statistical distribution with cost-based routing. For example, using Skill as a pseudo statistic makes sense, but using other properties, such as Customer Segment or Service Type for example, would not make sense.



Chapter

8

Solution Reporting

Genesys Stat Server provides URS with information in the form of statistics (see [Chapter 7](#)). In addition to the statistics being used for routing decisions, the statistics are also used for solution-specific reporting, such as when using the ERS and NRS reporting templates to create reports in CC Pulse+ and CC Analyzer.

In addition, Interaction Concentrator (ICON) works with T-Server and DB Server to collect and record calls and agent states in a database, which can be used to generate reports.

Starting with 7.6, Genesys Interactive Insights 7.6 provides reports that summarize the inbound voice-related activity that enters and is directed throughout your contact center.

This chapter overviews Genesys reporting as it relates to Universal Routing. It contains the following sections:

- [Reporting in a Typical Contact Center, page 741](#)
- [Solution Reporting, page 742](#)
- [Reporting Documentation, page 748](#)
- [Reporting Templates, page 749](#)
- [Reports From Interactive Insights, page 749](#)

Reporting in a Typical Contact Center

The processing of customer interactions is distributed over several elements of the contact center. For example, a typical inbound call from a customer can first be connected to an IVR to collect information about the interaction and/or the customer. Then, the call may be directed to a Universal Routing Server (URS), which finds the most appropriate agent, and finally be routed to an agent's desktop. Each of the various components involved in interaction processing and routing can be interesting from the reporting perspective.

Solution Reporting

Genesys offers several reporting applications:

- CCPulse+
- CC Analyzer
- Interaction Concentrator (ICON)

Each is briefly discussed below.

CCPulse+

CCPulse+ presents critical, business-related data in real time historically via on-screen reports. [Figure 258](#) shows an example voice report in CCPulse+.

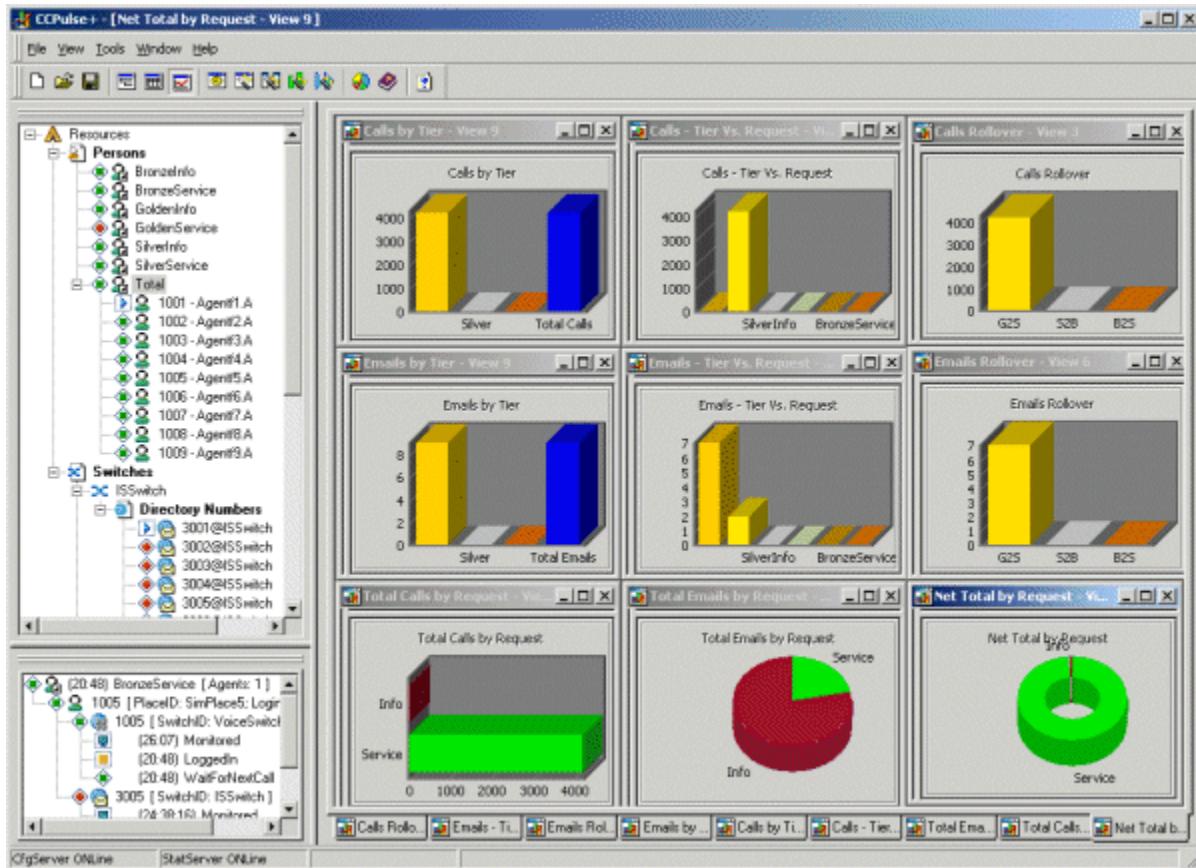


Figure 258: Sample CCPulse+ Real-Time Voice Report

See [Figure 259](#) for an example e-mail report.

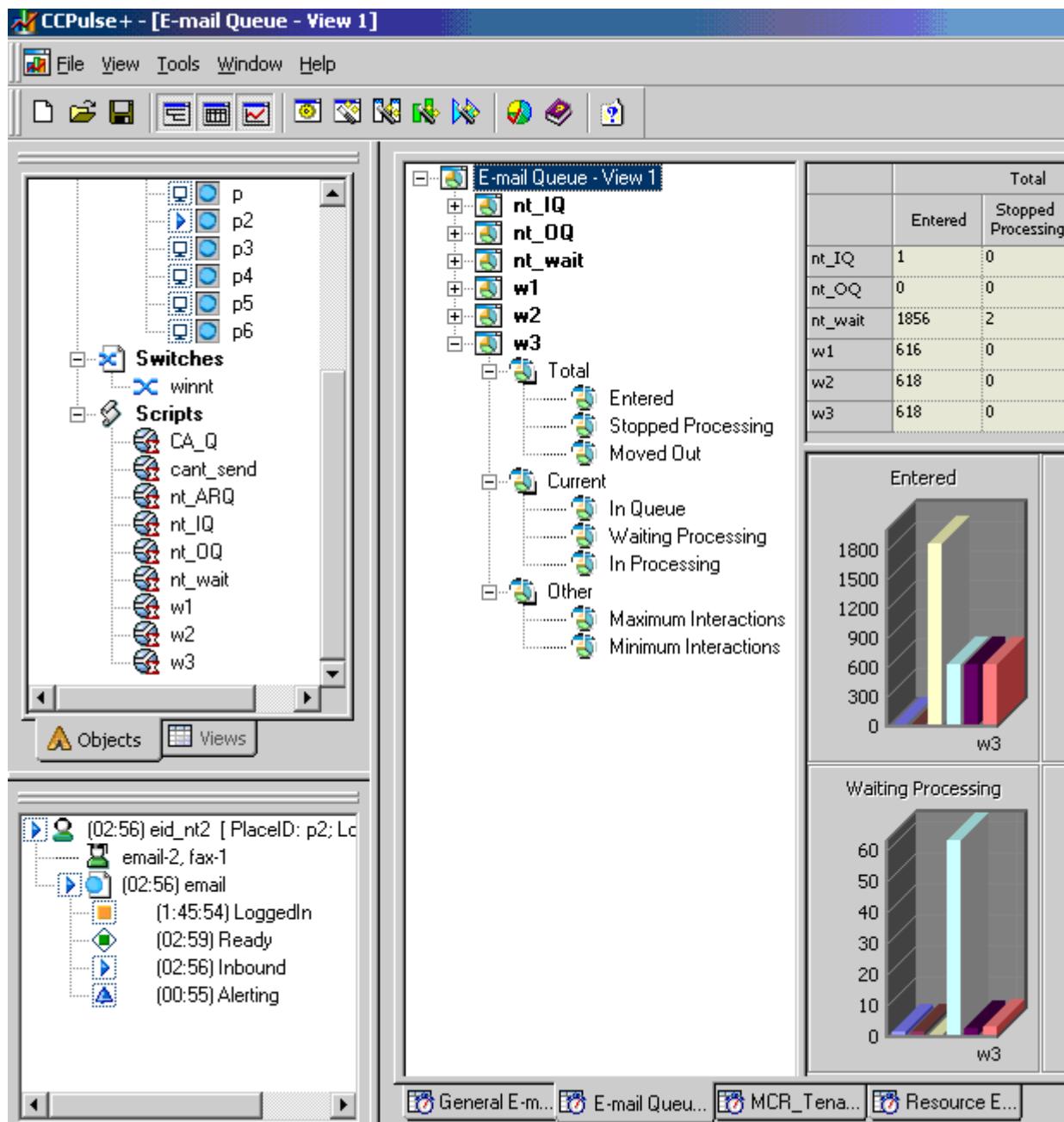


Figure 259: Sample CCPulse+ Real-Time E-mail Report Workspace

Note: When monitoring Contact Center activity using real-time reporting, limit statistics about Open Media interactions that use the Media Type Business Attribute in the StatType definition to single Media Types for each reportable object. The object types impacted by this are interaction queues, referred in StatType as “StagingArea,” and routing strategies.

Agent Readiness

The state of agent readiness in CCPulse+ and a capacity rule's state of readiness (see “use_agent_capacity” on [page 674](#)) differ.

- CCPulse+ reports Ready/Not Ready based on the state of a DN. If the agent has indicated that he is ready on a particular DN at a particular Place, then CCPulse+ displays the agent as Ready in its views.
- The capacity snapshot for a particular capacity rule, however, will indicate R (Ready) only if Universal Routing Server is permitted (from the definition of the capacity rule) to route an interaction to a particular DN.

Hence, it is possible, for example:

For an agent to have clicked the Ready button and be available to accept an interaction, but no interactions are sent.

This can be because the agent has exceeded the maximum number of concurrent interactions based on a capacity rule assigned to him.

CC Analyzer

The CC Analyzer application is an historical reporting application that collects and aggregates data over specified periods of time (see [Figure 260](#) for an example).

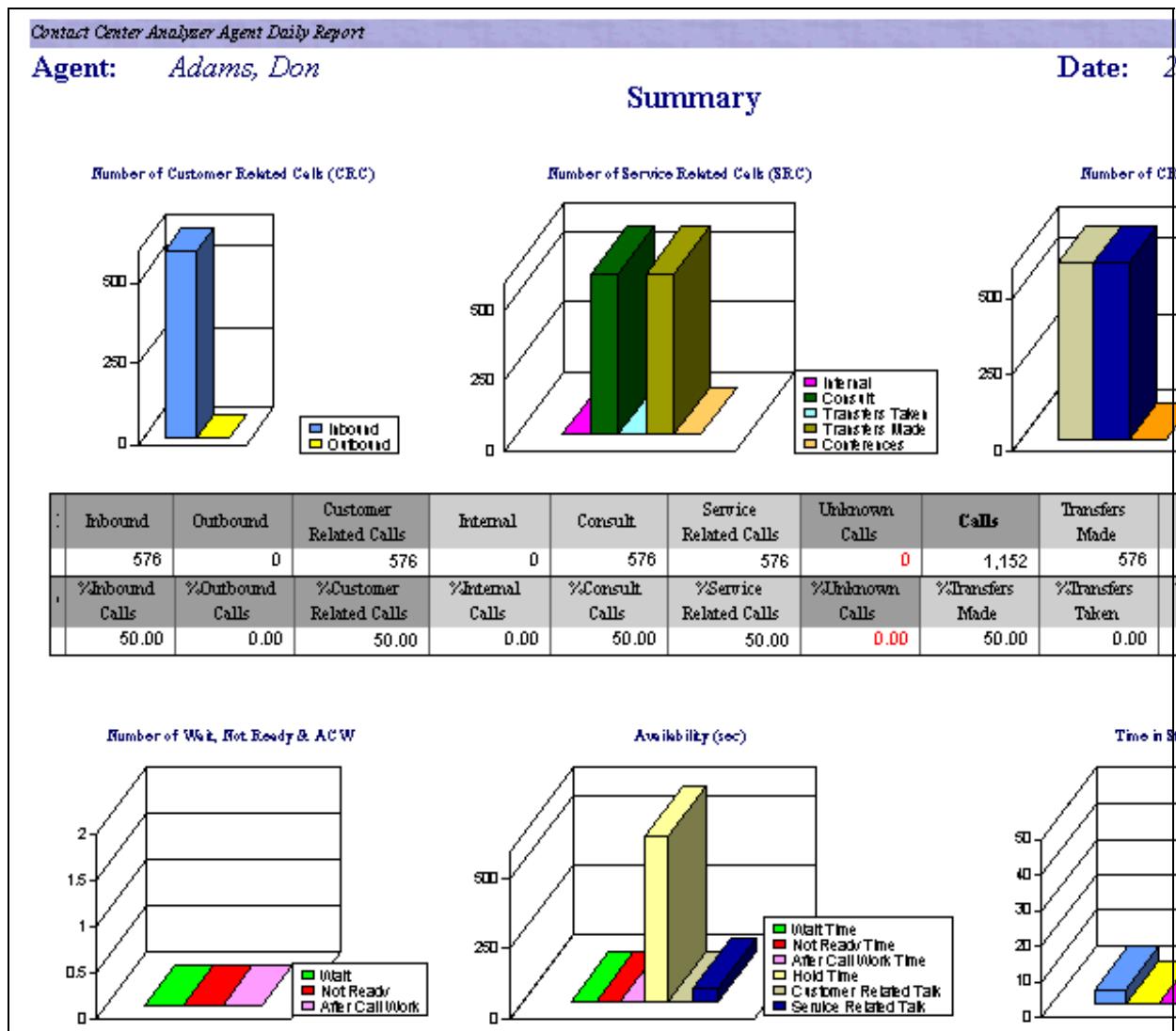


Figure 260: Sample CC Analyzer Historical Report

Interaction Concentrator

Interaction Concentrator (ICON) tracks historical data for your contact center as it relates to calls. This means reports are not based on the resources used during the call (CC Analyzer is better suited for that). Rather, the data collected by ICON can give you insight into the customer experience during the call.

ICON is a data collection application that works with T-Server and DB Server to collect and record calls and agent states. As a client of T-Server, ICON collects information about User Data and telephony events related to all telephony objects.

Note: ICON currently tracks only voice interactions.

Reporting on the Data

For reporting from ICON, Genesys provides the Genesys Info Mart, which produces a data mart containing several star schemas you can use for contact center historical reporting. Genesys Info Mart includes a software platform and a set of predefined jobs. You configure these jobs to extract and transform data from several Genesys relational databases. The transformed data is loaded into Dimension and Fact database tables in Genesys Info Mart. You can query the data in these tables using SQL. These queries enable you to display detailed data, reveal patterns, and predict trends.

Starting with 7.6, URS extends `EventDiverted` messages with a new `Reason` attribute that allows you to report on the reasons why calls leave virtual queues.

Reason Attribute

When URS receives an event message indicating a call has left a virtual queue, it copies the content of the `Reason` attribute into a `RequestDistributeEvent` message to T-Server, which causes T-Server to distribute the required event (`EventDiverted`).

URS receives reasons for calls leaving virtual queues from T-Server, which indicate that the routing process is terminated. Usually such events are `EventRouteUsed` or `EventReleased`.

- URS can receive these events as a result of successful routing (including default routing).
- URS can also receive these events as the result of the switch (or T-Server) taking over the call (such as during switch-level routing or revoking a call).

In these cases, the `Reason` attribute of the T-Server Event is propagated into an `RequestEventDiverted` message for virtual queues that URS distributes for such calls. In other cases, the reason the call left the virtual queue is unrelated to any T-Server event.

Extensions Attribute

If T-Server distributes the `EventDiverted` message with state 22, URS updates `AttributeExtensions` of the `EventDiverted` message with a new `Reason` key, which can contain one of the following values:

- 1 if the call is routed by URS to the target destination.
- 2 if the call is routed by URS to the default destination.
- 3 if the call is routed by the switch to the default destination.
- 4 if the call is cleared from a virtual queue by the execution of strategy function `ClearTargets`
- 5 is used for all other reasons.

Examples: The value of 5 is used if URS routes a call to an IVR destination and the IVR target has an associated virtual queue. The value of 5 is also used if a call is cleared because of a URS/T-Server disconnect or a routing point going out of service.

Note: When URS distributes virtual queue events, the URS log reflects this with RequestDistributeEvent messages to T-Server, which cause T-Server to distribute the required events.

An example RequestDistributeEvent message, with the new Reason key highlighted in red, is shown below.

```
request to 65200(--) message RequestDistributeEvent
    AttributeTimeout      10
    AttributeExtensions   [150] 00 06 00 00..
        'SIGNATURE'      'router'
        'NAME'            'vit_router'
        'VERSION'         'Version: 7.6.001.00'
        'CLUSTER'         'vit_router'
        'VQID'            '3J2L36BKDL77N4TKDU1MFDSAK0000003'
        'Reason'          4
    AttributeDNIS        '800'
    AttributeOtherDNRole 1
    AttributeOtherDN     '111111111111'
    AttributeThisDNRole 2
    AttributeThisQueue   '7778'
    AttributeThisDN     '7778'
    AttributeCallState   22
    AttributeCallType    2
    AttributeCallID     50
    AttributeConnID     006b017a7cecf028
    AttributeCustomerID 'Vit'
    AttributeReferenceID 4294967295
    AttributeUserEvent   EventDiverted
```

Cost-Based Routing and Reporting

Cost-based routing does not supply any “out-of-the-box” cost-based routing reports. Instead you can configure Genesys ICON/Info Mart to capture sufficient data from Universal Routing interaction attached data to allow the building of cost-based routing reports as a Genesys Professional Services engagement. For more information, see the *Universal Routing 7.6 Cost-Based Routing Configuration Guide*.

Reporting Documentation

As of the publication date of this manual, the following Reporting documents are available on the Genesys Documentation Library DVD:

- *Reporting Deployment Guide*
- *Reporting Data Sourcer User's Guide*
- *Reporting ETL Runtime User's Guide*
- *Reporting Physical Data Model for a Sybase Data Mart*
- *Reporting Physical Data Model for an Oracle Data Mart*
- *Reporting Physical Data Model for a Microsoft SQL Data Mart*
- *Reporting Physical Data Model for a DB2 Data Mart*
- *Reporting CCPulse+ Administrator's Guide*
- *Reporting Report Generation Assistant User's Guide*
- *Reporting Data Modeling Assistant Help File*
- *Reporting ETL Assistant Help*
- *Reporting CCPulse+ Help*
- *Reporting Reference Manual*
- *Reporting Master Index*

- *Genesys Reporting Technical Reference Guide for the Genesys Release*

For information on Interaction Concentrator, see the following documents:

- *Interaction Concentrator Deployment Guide*
- *Interaction Concentrator Physical Data Model for a DB2 Database*
- *Interaction Concentrator Physical Data Model for an Informix Database*
- *Interaction Concentrator Physical Data Model for a Microsoft SQL Database*
- *Interaction Concentrator Physical Data Model for an Oracle Database*

For information on Genesys Info Mart, see the following documents:

- *Genesys Info Mart Operations Guide*
- *Genesys Info Mart User's Guide*
- *Genesys Info Mart Reference* for your database management system

Reporting Templates

Reporting templates are available on the Reporting DVD, which contains ERS CCPulse+ reporting templates (.xml + storage) as well as CC Analyzer ERS reporting templates (.bqy).

Note: You can use 6.5 reporting templates in a 7.2 and later Universal Routing environment. If you do so, however, you can only create reports supported in 6.5. You cannot report on objects or functions introduced after 6.5.

Reports From Interactive Insights

Interactive Insights is the presentation layer that Genesys has designed for business-like interpretation of the source data that is collected and stored in the Genesys Info Mart database. Interactive Insights 7.6 provides various reports that summarize the inbound voice-related activity that enters and is directed throughout your contact center. The reports are summarized below.

Agent Reports

The Agent reports are as follows:

- Agent ACW Report
- Agent Group Inbound Call Handling Report
- Agent Group Service Type Report
- Agent Inbound Call Handling VQ Report
- Agent Inbound Utilization Report
- Agent Interval Based Report
- Agent Not Ready Reason Code Report
- Agent Not Ready Report
- Daily Agent Login-Logout Report
- Daily Agent State Detail Report

Queue Reports

The Queue reports are as follows:

- Abandon Delay Report
- Inbound Voice Traffic Group Report
- Inbound Voice Traffic Report

- Queue-Virtual Queue Summary Report
- Speed of Answer Report

Business Reports

The Business Reports are as follows:

- Call Volume Service Subtype Report
- Call Volume Service Type Report

Interactive Insights Documentation

The follow documents are available on the Genesys Documentation Library DVD:

- The *Genesys Interactive Insights 7.6 Deployment Guide*, which describes how to install Interactive Insights and the Genesys-provided image of Business Objects Enterprise XI, Release 3 on Windows and Unix platforms.
- The *Genesys Interactive Insights 7.6 Universe Guide*, which describes the elements referenced in the Interactive Insights reports as well as each element of the universe.
- The *Genesys Interactive Insights 7.6 User's Guide*, which describes how to customize the Interactive Insights universe and reports to perform extended functions.



Appendix

A Log Events

Universal Routing uses both Management Layer logging and logs specific to Universal Routing. For detailed information on Management Layer log events, see the “*Combined Log Events Help*,“ which is available on the [Genesys Documentation Website](#) or the Genesys Documentation Library DVD.

The `verbose` option configured in the Universal Routing Server (URS) Application object controls the logging level for Management Layer logging.

- For detailed information on Management Layer logging, refer to the *Framework 8.1 Deployment Guide* (Management Layer Functionality) or to the *Framework 8.1 Solution Control Interface Help* which are available on the [Genesys Documentation Website](#).

Warning! Do not confuse this option with other options having the same name, but located in different sections of this book. See the URS configuration option “`verbose`” on [page 684](#), and the user-defined `web` section for use by the Web Service object on [page 689](#).

The purpose of this appendix is to provide an overview of Universal Routing log events and to list the specific log message text used for the `verbose` option, described on [page 684](#).

This Appendix includes the following sections:

- [Management Layer Log Output Levels, page 752](#)
- [Universal Routing Log Messages, page 753](#)

Also see the information on the function “`Alarm`” on [page 506](#).

Management Layer Log Output Levels

Genesys applications, such as URS, can report log events at five levels of detail:

- Alarm
- Standard
- Interaction
- Trace
- Debug

Only the first four are intended for on-site analysis by a user. Log events of the Alarm, Standard, Interaction, and Trace levels feature the same unified log record format, can be stored in the Centralized Log Database, and are viewable via the Solution Control Interface (SCI).

Logging During Normal Operation

Logging levels used for normal operation are Alarm, Standard, Interaction, and Trace.

Alarm Log Level

The Alarm-level logs contain only log records of the Alarm level. Solution Control Server (SCS) generates Alarm log events on behalf of other applications when receiving from them log events configured as Detection Events in Alarm Conditions. Using this level, Solution Control Server reports the occurrence and removal of all alarms to the Centralized Log Database.

This level contains the Management Layer translations of log events of other levels into Alarms.

Standard Log Level

Permanently enable only the Standard level of logging during solution operation in regular production mode. This level generates high-level events that report both major problems and normal operations of in-service solutions.

Interaction Log Level

The Interaction-level logs report the details of an interaction processed by Solution components that handle interactions. The log contains information about the processing steps for each interaction by each Solution component.

Trace Log Level

The Trace-level logs report the details of communications between the various Solution components. The log contains information about the processing steps for each interaction by each Solution component.

Logging During Irregular Operation

Standard-level, Interaction-level, and Trace-level log events do not contain all the details you or someone else may need to analyze and troubleshoot solutions malfunctions. That's why Genesys Customer Care might ask you to provide relevant Debug-level logs when you request their assistance.

Because the Debug-level log events do not have a unified format, are not documented, and can only be stored in a text file, they are only useful to Genesys Customer Care. Logging at this level is likely to adversely affect application performance. Enable this log output level only when a Genesys representative requests it.

Note: For more information on the levels, refer to the *Framework 8.1 Deployment Guide*.

Log Filtering

The [log-filter] and [log-filter-data] sections contain configuration options used to define the treatment of filtering data in log output on a key-by-key basis. For complete information of configuration options defined in these sections refer to chapter “Hide Selected Data in Logs” in the *Genesys 8.1 Security Deployment Guide*.

Universal Routing Log Messages

Note: This appendix supplements information found in the verbose option description on [page 684](#).

You can configure the verbose option (see [page 684](#)) in the URS Application object’s default section to specify which log messages you want to have written to the URS log.

Each log message is associated with a number from 1-5. Set the verbose option to the number that corresponds to the messages you want to see in the log. Setting a higher value means that all messages with that value or lower appear in the log. That is, if you set the value to 3, messages marked 1, 2, and 3 appear in the log.

Changing Log Message Values

Starting with release 7.2, you can change the number associated with each log message. For example, if you want to set the verbose option to 3, but also want to see a message type currently marked as level 5, you can change the message type level from 5 to 3. This prevents you from having to set the value to 5 in order to get the level 5 message type you want and then having to sort through all the rest of the level 4 and 5 messages that you are not interested in using.

To customize the verbose level of a message, you must know its message ID. Every message has a unique identifying (ID) number.

To change message verbose level, create the `VerboseLevels` folder in the URS Application object. The content of this folder is a series of key-value pairs, in which the key is the message ID and the value is the message's verbose level.

A list of all URS log messages with their IDs and default verbose levels appears in the next section.

URS Log Message List

In order to help you better understand the `verbose` option, the remainder of this appendix groups the messages based on their verbose level.

```

001: level=1,      text=can't find server with socket %d
002: level=1,      text=server %s is being closed, socket %d
003: level=1,      text=bad packet(length %d) for server %s
004: level=1,      text=can't register handler for client %d
005: level=1,      text=call is reassigned to tenant %s
006: level=1,      text=call with no tenant
007: level=1,      text=call (%ld-%p) for %s created (del %ld)
008: level=1,      text=call (%ld-%p) cloned from %p
009: level=1,      text=strategy: %s is attached to the call
010: level=1,      text=call collision (%s)
011: level=3,      text=treatment added: type=%s, param1=%s, param2=%s, tout=%d
012: level=3,      text=treatment moved: type=%s, param1=%s, param2=%s, tout=%d
013: level=3,      text=treatment deleted: type=%s, param1=%s, param2=%s, tout=%d
014: level=1,      text=call deleting %s
015: level=1,      text=unknown result to process
016: level=2,      text=call waits too long (%ld sec)
017: level=1,      text=look for hanged interactions: %d at all now
018: level=1,      text=there are %d calls in progress now
019: level=1,      text=I AM ALMOST DEAD (%p,%c in %d)
020: level=1,      text=exception happens
021: level=1,      text=port for %s coincides with its management port, the last will be ignored
022: level=1,      text=can't initialize management layer
023: level=1,      text=Initialization completed
024: level=1,      text=Application started
025: level=1,      text=Application initialization failed, reason %s
026: level=1,      text=Normal termination
027: level=1,      text=Application terminated due to internal condition

```

```

028: level=1, text=Application stopped
029: level=1, text=change run mode request (%s -> %s)
030: level=4, text=unknown option %s for server %
031: level=3, text=invalid value %s for option %s for server %
032: level=1, text=unknown type %s for server %s, skipped
033: level=4, text=Configuration option changed: %s:%s = %
034: level=4, text=Configuration option set: %s:%s = %
035: level=4, text=Configuration option set to default value: %s:%s = %
036: level=1, text=start to read configuration data from file %
037: level=1, text=read configuration data from file %s - success
038: level=1, text=read configuration data from file %s - failure
039: level=1, text=open configuration for tenant %ld
040: level=1, text=close configuration for tenant %ld
041: level=1, text=get %s%ss for tenant %ld
042: level=4, text=OK InfoMessage is received from server %
043: level=2, text=error in cInfoMessage from server %s, call is in status CALL_STATUS_XDATA
044: level=3, text=unhandled request is deleted
045: level=1, text=Exception is received for custom server %
046: level=2, text=%sdata lookup failed
047: level=2, text=MSG_ERROR is received from dbserver %s, call is in status CALL_STATUS_XDATA
048: level=2, text=MSG_ERROR is received from dbserver %s for table access %
049: level=4, text=MSG_RETRIEVED(status success) is received from dbserver %
050: level=4, text=MSG_RETRIEVED(status success) is received from dbserver %s for table access %
051: level=3, text=MSG_RETRIEVED(status nomore or empty) is received from dbserver %
052: level=3, text=MSG_RETRIEVED(status nomore or empty) is received from dbserver %s for table
    access %
053: level=2, text=MSG_RETRIEVED with unknown status %d is received from dbserver %s,
    call_status=CALL_STATUS_XDATA
054: level=2, text=MSG_RETRIEVED with unknown status %d is received from dbserver %s for table
    access %
055: level=3, text=MSG_PROCCOMPLETED (status %d) is received from dbserver %
056: level=3, text=unhandled request is deleted
057: level=3, text=unhandled table access (%s) request is deleted
058: level=1, text=DOpenDatabase fails
059: level=3, text=request %ld to dbserver %s sent: %
060: level=3, text=event waiting time is over
061: level=1, text=EventAnswerAccessNumber is received from tserver %
062: level=1, text=EventAgentReserved is received from tserver %
063: level=1, text=Agent reserving is out of time and canceled
064: level=1, text=AgentReservation failed (tserver %s)
065: level=1, text=<-----EXTROUTER
066: level=3, text=unhandled requests is deleted
067: level=1, text=make request address for switch %s (tserver %s)
068: level=1, text=fail to request for address for switch %s (tserver %s)
069: level=1, text=answer time for extrouter is over
070: level=1, text=send to tserver %s TReserveAgent (dn=%s, name=%s, place=%s, priorities=%d,%d,%d)
071: level=3, text=double route call request
072: level=3, text=targets list is empty
073: level=3, text=wait time is over
074: level=4, text=start chain of treatments
075: level=5, text=delay treatments for %d msec

```

```

076: level=3, text=invalid call state %d
077: level=3, text=HERE IS WAIT FOR DN %d (%d sec)
078: level=3, text=update attached statistics
079: level=3, text=timeout is over
080: level=3, text=HERE IS XDATA
081: level=4, text=error in strategy: %s
082: level=3, text=fail to make DB server request
083: level=3, text=wait IVR time is over
084: level=3, text=HERE IS WAIT_IVR_DELIVERY
085: level=1, text=play treatment end=====>TREATMENT_NONE
086: level=3, text=HERE IS PLAY TREATMENT
087: level=3, text=HERE IS START TREATMENT
088: level=3, text=HERE IS WAIT (%ld sec)
089: level=3, text=wait treatment end time is over
090: level=3, text=HERE IS WAIT_TREATMENT_END(%d sec)
091: level=3, text=there is return value, continue
092: level=3, text=continue Ok
093: level=3, text=continue fail
094: level=3, text=pause (%ld msec)
095: level=3, text=pause is over
096: level=3, text=HERE IS TARGETS
097: level=3, text=current service factor=%d, working set=%d
098: level=3, text=||||||>>>>continue business rule(%d)
099: level=3, text=||||||<<<<suspend business rule
100: level=3, text=business rule is about to complete
101: level=3, text=HERE IS BUSINESS RULE %s
102: level=3, text=HERE IS WORKFORCE SCHEDULE %s
103: level=3, text=expression translating: %s -> %s
104: level=3, text=expression translating(%s) fails
105: level=3, text=skill %s for group %s
106: level=3, text=HERE IS ROUTE TO TARGET: %s
107: level=3, text=target <%s> is not valid or not current
108: level=3, text=HERE IS DELIVER TO TARGET(%d): %s
109: level=3, text=check list(%s) %s(SQL: %s) for %s
110: level=3, text=list data received: %s
111: level=3, text=check cfglist(%s) for item %s, result: %d
112: level=3, text=get data from cfglist(%s) item %s, key %s result: %s
113: level=3, text=jump to strategy %s
114: level=3, text=call strategy %s (level=%d, cnt=%d from %ld.%0.3ld)
115: level=3, text=return to strategy %s (level=%d)
116: level=3, text=subroutine parameters mismatch: %d
117: level=3, text=HERE IS ANSWER CALL
118: level=3, text=fail to answer call on DN %s
119: level=3, text=HERE IS RELEASE CALL
120: level=3, text=fail to release call on DN %s
121: level=5, text=no error mode for this call
122: level=1, text=OP_RETURN: invalid operand type: %d
123: level=1, text=OP_JTRUE: invalid operand type: %d
124: level=1, text=OP_JFALSE: invalid operand type: %d
125: level=0, text=mission impossible(%c%c%c%c), go to default
126: level=1, text=OP_XCALL: invalid operand type: %d

```

```

127: level=2, text=OP_XCALL: function %s returns INTERP_ERROR, no error handling
128: level=2, text=OP_XCALL: function %s returns INTERP_ERROR, go to error handling
129: level=2, text=OP_XCALL: function %s returns INTERP_STOP, no stop handling
130: level=2, text=OP_XCALL: function %s returns INTERP_STOP, go to stop handling
131: level=1, text=OP_XCALL: function %s returns unknown code
132: level=1, text=OP_XCALL_NO_RESULT: invalid operand type: %d
133: level=2, text=OP_XCALL_NO_RESULT: function %s returns INTERP_ERROR, no error handling
134: level=2, text=OP_XCALL_NO_RESULT: function %s returns INTERP_ERROR, go to error handling
135: level=2, text=OP_XCALL_NO_RESULT: function %s returns INTERP_STOP, no stop handling
136: level=2, text=OP_XCALL_NO_RESULT: function %s returns INTERP_STOP, go to stop handling
137: level=2, text=OP_XCALL_NO_RESULT: function %s returns unknown code
138: level=1, text=out of interpreter loop
139: level=1, text=interpreter pointer is out of range
140: level=1, text=resume interp with bad opcode %d
141: level=2, text=resume interp with INTERP_ERROR state, go to error handling
142: level=2, text=resume interp with INTERP_ERROR state, no error handling
143: level=2, text=resume interp with INTERP_STOP state, go to stop handling
144: level=2, text=resume interp with INTERP_STOP state, no stop handling
145: level=1, text=resume interp with unknown state
146: level=1, text=>>>>>>>>start interpreter(%s)
147: level=1, text=fail to interp, see above why, else stack problem
148: level=1, text=>>>>>>>>resume interpreter(%d), func:%s
149: level=1, text=<<<<<<<<suspend interpreter(%s), func:%s timers:%d%d%d%d
150: level=1, text=<<<<<<<<stop interpreter after error
151: level=1, text=<<<<<<<<stop interpreter
152: level=1, text=Licensing violation identified, violation type %s
153: level=1, text=Feature %s: %d licenses checked out
154: level=1, text=Licensing violation condition, violation type %s, no longer exists
155: level=4, text=converting tenant:%s name:%s to switch:%s, dn:%s
156: level=6, text=option %s=%c%s%c
157: level=6, text=there is no any value for option %s
158: level=4, text=can not find application for tserver %s
159: level=4, text=can not find switch for tserver application %s
160: level=2, text=reread configuration data from delta file
161: level=3, text=check business rule %s quality (%d entries, %d delta)
162: level=3, text=business rule %s violation for %s criteria: %hd level - %hd% ( %hd% )
163: level=3, text=business rule %s AWT - %hd(%hd) sec
164: level=3, text=current agent number for business rule %s: %d
165: level=4, text=bad prognosis: %d - %d
166: level=4, text=exit routing rule: success(%d) stat_ids(%hd %hd) entry(%ld %hd) entries(%hd)
167: level=4, text=translation of type [TSERVER], can't find tserver for switch %s
168: level=4, text=translation of type [AGENT....], there is no agent name
169: level=4, text=translation of type [PLACE....], there is no place name
170: level=4, text=translation of type [AGENT....], can't find tenant %s
171: level=4, text=translation of type [AGENT....], can't find agent %s
172: level=4, text=translation of type [AGENT....], can't find login for agent %s on switch %s
173: level=4, text=translation of type [DN.DL], can't find group (%d) for dn %s
174: level=4, text=translation of type [DN.DL], empty translation group %
175: level=4, text=translation of type [TARGET.CCTN], there is no target name
176: level=4, text=translation of type [TARGET.CCTN], unsupported target type
177: level=3, text=translating(%d) results in NULL string

```

```

178: level=3, text=invalid substitution code %
179: level=4, text=can't find destination switch %
180: level=4, text=there is no switch access codes from server %s to switch %
181: level=3, text=fail to translate destination number %s, fail to route call
182: level=3, text=fail to define route type for dn %s, RouteTypeUnknown will be used
183: level=3, text=TRANSLATED(%s): %s@%s -> route_type = %d, destination_label = %s, dnis = %s,
    destination_location = %s
184: level=3, text=NO TRANSLATION for this call
185: level=3, text=tserver %s is not open
186: level=3, text=can not attach data
187: level=1, text=fail to send call, reason %
188: level=1, text=----->ARRIVAL
189: level=1, text=----->DEPARTURE(processing time %ld)
190: level=1, text=<-----ERROR(fail to continue sending)
191: level=1, text=call have no location
192: level=1, text=----->EXTROUTER
193: level=1, text=<-----TIMER
194: level=1, text=----->TIMER
195: level=1, text=----->TMESSAGE
196: level=1, text=HERE IS ROUTE_CALL: VQ=%s, target=%s
197: level=1, text=route call from invalid state %d
198: level=1, text=HERE IS DELIVER_CALL
199: level=2, text=HERE IS ROUTE_IVR
200: level=1, text=route ivr from invalid state %d
201: level=1, text=route ivr from invalid treatment state %d
202: level=1, text=IVRTreatment was continued=====>TREATMENT_REQUEST
203: level=1, text=WaitIVRDelivery from invalid call status
204: level=1, text=Call delivered on IVR Ok
205: level=1, text=Call delivered on IVR fail
206: level=1, text=Call delivered, wait IVR notification
207: level=3, text=routing done
208: level=3, text=continue routing process
209: level=1, text=call will be deleted due to this error
210: level=1, text=emergency: delete call due to this error
211: level=1, text=emergency: ignore this error
212: level=1, text=emergency: send call to default due to this error
213: level=1, text=emergency: reroute call due to this error
214: level=1, text=emergency: continue strategy on this error
215: level=1, text=emergency: go to error handling on this error
216: level=2, text=HERE IS ROUTE_DEFAULT%
217: level=2, text=HERE IS ROUTE_DIRECT
218: level=4, text=check call routing states: state=%d delivery=%d treatment=%d held=%d reserving=%d
    ivr=%d - %s
219: level=2, text=HERE IS ABANDONED%
220: level=1, text=Cannot connect to %s [%s] at host [%s], port %d, reason %
221: level=1, text=new backup server for %s is found: %s, replace first server by second one
222: level=1, text=Connecting to %s [%s] at host [%s], port %d
223: level=1, text=Server [%s] contacted, socket %d
224: level=1, text=Connected to %s [%s] at host [%s], port %d
225: level=1, text=server of unknown type %d will be created - %
226: level=1, text=Connection to %s [%s] at host [%s], port %d lost

```

```

227: level=1, text=change %s current server from %s to %s
228: level=1, text=Disconnected from %s [%s]
229: level=1, text=server %s is deleting
230: level=1, text=abort request time is over
231: level=1, text=SERVER %s (id %p, class %d, type %s, parent %p) created
232: level=1, text=SERVER %s (id %p, class %d, type %s, parent %p) deleted
233: level=1, text=SERVER %s(%p) is current for %s(%p)
234: level=3, text=target STATISTIC mapping error
235: level=3, text=target STATISTIC N/A
236: level=3, text=input data error
237: level=3, text=statistic is not specified
238: level=3, text=HERE IS SDATA: %s@%s - %s
239: level=3, text=can not find stat server %s
240: level=3, text=can not find stat object %s[%s] for stat server %s
241: level=3, text=sdata returns %.12g(%c)
242: level=3, text=can not get sdata, 0 will be used
243: level=3, text=expand group returns(%c): %s
244: level=3, text=unknown target type: %d
245: level=5, text=attempt to route (%d)
246: level=3, text=default priority %d
247: level=3, text=priority tuning: age of interaction=%s, prediction=%s, service objective=%s
248: level=3, text=current priority for %s VQ(s): %d
249: level=3, text=current priority for %s VQ(s) increased by %d
250: level=3, text=percentage object was not found, creation
251: level=3, text=percentage object for target %s was not created
252: level=3, text=blocking: agent %s, place %s, switch %s dn %s is blocked for %ld sec by %ld, will
     expire %ld
253: level=3, text=agent %s, place %s, switch %s dn %s is already blocked
254: level=3, text=object %s is unblocked
255: level=3, text=place %s is not in pool of available seats: %d (max %d) are currently used
256: level=3, text=agent %s, place %s, switch %s dn %s isn't blocked absolutely(%ld)
257: level=3, text=agent %s, place %s is blocked absolutely
258: level=3, text=can not find switch or number for Queue <%s>
259: level=3, text=statistic <%s> for object <%s> closed
260: level=3, text=object VALID event: statistic <%s> object <%s>
261: level=3, text=object INVALID event: statistic <%s> object <%s>
262: level=3, text=stat server ERROR <%s>
263: level=3, text=stat server statistic error: server <%s>, statistic <%s>, object <%s>
264: level=0, text=event info (object <%s>, type <%s> statistic <%s> server <%s> value <%s>)
     request %ld is received
265: level=3, text=STATOBJECT(%p %ld %d) tenant=%s name=%s%s%s@s%.%s: content updated #%ld %
266: level=3, text=check queue for statobject <%s> type <%s>
267: level=5, text=state of call isn't good for routing
268: level=5, text=try to route to agent "%s" (place "%s", %d ready DNs reported): Target %p VQ %p
269: level=5, text=tenant %s queue %s: threshold value %.12g (compare to %d)
270: level=5, text=try to route to queue %s: Target %p VQ %p
271: level=5, text=queue %s not ready
272: level=5, text=dn %s@%s not ready
273: level=5, text=agent %s, place %s, switch %s dn %s is blocked with call %ld (%d sec from %ld)
274: level=3, text=LAUNCHED (dn=%s)
275: level=5, text=fail to launch (dn=%s)

```

```

276: level=1,    text=PULSE (calls: %d(%d)=%d+%d-%d, targets=%d, time=%ld, mem=%d,%lu,%lu,%lu,%lu,%lu)
277: level=3,    text=PULSE too long (%d sec)
278: level=5,    text=tenant %s %s %s: dn %s(ready since %ld) is OK
279: level=5,    text=tenant %s %s %s: dn %s (%c %d) is ignored: %
280: level=4,    text=reset ready DNs for object <%s> type <%s>
281: level=4,    text=ready DN(switch %s, number %s, type %d, time=%ld) for agent %s, place %s, status
               %s time=%ld
282: level=4,    text=ready media skill(ID=%d, name=%s) for AP %s (loading= %d)
283: level=3,    text=AP %s Loading= %d
284: level=4,    text=media skill(ID=%d, name=%s) for AP %s loading= %d
285: level=4,    text=states for agent %s: %x
286: level=3,    text=STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: %d ready DNs reported, dT=%d
287: level=3,    text=STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: %d ready media skills reported,
               loading=%d
288: level=3,    text=STATOBJECT(%p %ld %d) tenant %s name=%s@%s.%s: %sready
289: level=3,    text=STATOBJECT(%p %ld %d) tenant %s name=%s@%s.%s: no media skill %
290: level=3,    text=STATOBJECT(%p %ld %d) tenant %s name=%s@%s.%s: CHANGE OF STATE (%s->%s)
291: level=3,    text=STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: statistic <%s> created
292: level=3,    text=STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: statistic <%s> deleted
293: level=4,    text=STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: statistic <%s> failed
294: level=4,    text=STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: statistic <%s> asked
295: level=6,    text=modelling for statistic <%s> is not supported
296: level=3,    text=current number of statistics opened <%d>
297: level=3,    text=statistics deleted on last iteration <%d>
298: level=3,    text=can not find statserver for tserver %
299: level=3,    text=there is no threshold value for CDN %
300: level=3,    text=there is no any information for DN %
301: level=3,    text=%d is assumed to be parked
302: level=5,    text=STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: not assigned to workforce activity
               %
303: level=3,    text=current number of targets for tenant %s: Agents %d, Places %d, AgentGroups %d,
               PlaceGroups %d, ACDQueues %d, Routing Points %d
304: level=4,    text=STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: there is no config object
305: level=3,    text=initiate content updating process for all groups statistics due to update of some
               %s (%d total)
306: level=1,    text=SEventServerDis(Not)connected is received for statserver %
307: level=3,    text=current virtual queue: %p id=%d, nVQ=%s, priority=%d, time=%ld.%ld
308: level=3,    text=call (virtual queue %p, id=%d, priority %d) doesn't wait for VQ %p (name="%s")
               now
309: level=3,    text=call (virtual queue %p, id=%d, priority %d, time %ld.%ld) waits for VQ %p
               (name="%s") now
310: level=3,    text=entering virtual queue "%s"
311: level=0,    text=mission impossible 3 for call (virtual queue %p "%s", id=%d, priority %d, VQ %p)
312: level=3,    text=event %s for virtual queue %s is pending
313: level=2,    text=can not find virtual queue %s for tenant %s(%ld), virtual queue support is
               ignored
314: level=2,    text=can not find switch for virtual queue %s for tenant %s(%ld), virtual queue
               support is ignored
315: level=2,    text=can not find tserver for virtual queue %s for tenant %s(%ld), virtual queue
               support is ignored
316: level=1,    text=router starts activation (backup level=%d), router has waited %d sec

```



```

317: level=1, text=router is forced into active mode
318: level=1, text=router is selfactivated (backup level=%d), time - %d sec
319: level=1, text=router is forced into backup mode
320: level=1, text=router is forced into backup mode with another router of level %d
321: level=2, text=call% is lost
322: level=2, text=check delayed proc
323: level=1, text=execute delayed action
324: level=1, text=send to tserver TCompleteTransfer
325: level=1, text=send to tserver TCompleteConference
326: level=1, text=can not create or validate call for tserver %s, reject routing
327: level=1, text=add DN %s %s <%s@%s> (%s %d %s) to the call %ld-%p %s:%c%c
328: level=1, text=cancel selfdestruction
329: level=1, text=del DN (%s %s) %s(ref.id=%d)
330: level=2, text=there is no DNs for call, %s
331: level=1, text=EventQueued is received for tserver %s (this dn=%s)
332: level=1, text=EventRinging is received for tserver %s (this dn=%s)
333: level=1, text=EventEstablished is received for tserver %s (this dn=%s)
334: level=1, text=EventRouteRequest is received for tserver %s (this dn=%s)
335: level=1, text=EventPartyChanged is received for tserver %s (this dn=%s): %s -> %s
336: level=1, text=EventRouteUsed(%s) is received for tserver %s (this dn=%s)
337: level=1, text=EventReleased is received for tserver %s (this dn=%s)
338: level=1, text=EventDiverted is received for tserver %s (this dn=%s)
339: level=2, text=call is on hold
340: level=2, text=call is retrieved
341: level=1, text=<-----ARRIVAL
342: level=1, text=<-----DEPARTURE
343: level=1, text=<-----TMESSAGE
344: level=1, text=wandering call come to cdn
345: level=1, text=resume routing process
346: level=1, text=terminate current routing and start new one from the beginning (%d %d)
347: level=2, text=%scall is routed to target %s
348: level=2, text=Let's go%
349: level=1, text=threshold gate is closed, route to default
350: level=1, text=start without strategy -> %s
351: level=1, text=EventAbandoned is received for tserver %s (this dn=%s), %s call
352: level=1, text=EventPrivateInfo is received for tserver %s (this dn=%s)
353: level=1, text=EventTreatmentApplied is received for tserver %s (this dn=%s)
354: level=1, text=EventTreatmentNotApplied is received for tserver %s (this dn=%s)
355: level=1, text=EventTreatmentEnd is received for tserver %s (this dn=%s)
356: level=1, text=EventTreatmentRequired is received for tserver %s (this dn=%s)
357: level=1, text=EventDialing is received for tserver %s (this dn=%s)
358: level=1, text=EventError is received for tserver %s - %s
359: level=1, text=EventDestinationBusy is received for tserver %s (this dn=%s)
360: level=1, text=EventPartyInfo is received for tserver %s (this dn=%s)
361: level=1, text=<-----ERROR
362: level=3, text=Unable to register DN [%s], reason: %s
363: level=2, text=EventAttachedDataChanged is received for tserver %s (this dn=%s)
364: level=2, text=EventDigitsCollected is received for tserver %s (this dn=%s)
365: level=1, text=EventLinkConnected is received for tserver %s, register all CDNs and IVRs again
366: level=1, text=EventServerConnected is received for tserver %s
367: level=1, text=EventRegistered is received for tserver %s (this dn=%s)

```

```

368: level=1, text=EventServerDisconnected is received for tserver %
369: level=1, text=EventLinkDisconnected is received for tserver %
370: level=1, text=EventServerInfo is received for tserver %
371: level=1, text=EventLocationInfo is received for tserver %
372: level=1, text=EventDNOutOfService is received for tserver %s (this dn=%s)
373: level=1, text=EventDNIOutOfService is received for tserver %s (this dn=%s)
374: level=1, text=calls for %s %s will be deleted
375: level=3, text=register address: %s
376: level=1, text=send to tserver %s RequestRouteCall to dn %s on %s (dnis= %s)
377: level=1, text=send to tserver %s RequestMakePredictiveCall to dn %s
378: level=1, text=send to tserver %s TSingleStepTransfer to dn %s on %s
379: level=1, text=send to tserver %s TMuteTransfer to dn %s on %s
380: level=1, text=send to tserver %s TInitiateTransfer to dn %s on %s
381: level=1, text=send to tserver %s TInitiateConference to dn %s on %s
382: level=1, text=send to tserver %s TAnswerCall on dn %s
383: level=1, text=send to tserver %s TReleaseCall on dn %s
384: level=1, text=send to tserver %s TRequestPrivateService on dn %s: server %s, type %d, service
    %s, method %s
385: level=1, text=treatment end=====>TREATMENT_NONE
386: level=1, text=treatment applied=====>TREATMENT_APPLIED
387: level=1, text=treatment not applied=====>TREATMENT_NONE
388: level=2, text=%s% treatment %s fail
389: level=1, text=request to abort treatment while waiting IVR number=====>TREATMENT_NONE
390: level=1, text=request to abort treatment while treatment is requested, reject
391: level=1, text=request to abort treatment while treatment is still requested, reject
392: level=1, text=request to abort treatment after treatment is applied=====>TREATMENT_NONE
393: level=1, text=request to abort treatment in unknown state, reject
394: level=1, text=treatment timer is activated in ASK_NUMBER state
395: level=1, text=treatment timer is activated in REQUEST state=====>TREATMENT_AFTER_TIME
396: level=1, text=treatment timer is activated in APPLIED state
397: level=1, text=treatment timer is activated in bad treatment state: %d
398: level=1, text=TGiveMusicTreatment was applied
399: level=1, text=TGiveRingBackTreatment was applied
400: level=1, text=TGiveSilenceTreatment was applied
401: level=1, text=PauseTreatment was applied
402: level=1, text=IVRTreatment was started=====>TREATMENT_ASK_NUMBER
403: level=1, text=fail to make IVRTreatment=====>TREATMENT_NONE
404: level=1, text=compatible treatment=====>TREATMENT_APPLIED
405: level=1, text=treatment starting: type=%s, param1=%s, param2=%s, tout=%d
406: level=2, text=treatment starting - ok
407: level=1, text=apply treatment when treatment_status is not NONE
408: level=1, text=move call home before treatment starting
409: level=1, text=fail to move call home before treatment starting
410: level=1, text=failed to find default label for IVR treatment for cdn %s
411: level=1, text=TApplyTreatment (type %s) is being called
412: level=1, text=treatment=====>TREATMENT_REQUEST
413: level=1, text=unknown treatment - %d
414: level=3, text=current number of interactions per second - %.2f
415: level=3, text=current number of entries: for longest queue %d, for all queues %d
416: level=4, text=tmessage
417: level=1, text=Exception is received for message server %s

```

```

418: level=6,    text=monitoring message (timeout %d of %d, %s) was sent, server %s, socket %d
419: level=3,    text=VQ %p (virtual queue "%s", id=%d), (%d Targets): SELECT %s by statistic <%s>(%s
    %s %s)
420: level=3,    text=VQ %p Target "%s"(%p) #%d, (%d-%d Components): SELECT %s by statistic <%s>(%s
    %s): MEM(%ld %ld) %d %d
421: level=5,    text=VQ %p Target "%s"(%p) PERCENTAGE mapping error
422: level=5,    text=VQ %p Target "%s"(%p): statvalue: %d, %.12g(%c) %s
423: level=5,    text=VQ %p Target "%s"(%p): statvalues: %d, %d %d %d(%c) %s
424: level=5,    text=VQ %p Target "%s"(%p): empty(%d-%d components) passed
425: level=5,    text=VQ %p Target "%s"(%p): not ready passed
426: level=5,    text=VQ %p Target "%s"(%p) Component #%d %s: queue is hold by call %s
427: level=5,    text=VQ %p Target "%s"(%p) Component #%d %s: statvalue: %d, %.12g(%c) %s
428: level=5,    text=VQ %p Target "%s"(%p) Component #%d %s: statvalues: %d, %d %d %d(%c) %s
429: level=5,    text=VQ %p Target "%s"(%p) Component #%d %s: %s
430: level=3,    text=VQ %p (virtual queue "%s" id=%d): Target "%s"(%p) for route was SELECTED
431: level=3,    text=VQ %p Target "%s"(%p): Component %s was SELECTED: MEM(%ld) %d %d
432: level=3,    text=VQ %p (virtual queue "%s" id=%d): Target for routing was NOT SELECTED (%d %d %d %d
    %d)
433: level=3,    text=VQ %p Target "%s"(%p): Component for routing was NOT SELECTED (%d %d %d %d)
434: level=5,    text=VQ %p is hold by call %s
435: level=5,    text=VQ %p check threshold: %s => %d
436: level=5,    text=VQ %p Target "%s"(%p): out of threshold limits
437: level=5,    text=VQ %p is ignored: %s
438: level=3,    text=VQ %p Target %p added: name=%s, location=%s, type=%s, state=%s, activity=%s
439: level=3,    text=VQ %p created: type=%d, tenant=%s
440: level=3,    text=VQ %p deleted
441: level=3,    text=VQ %p [at all %d %d] %d Target(s), flag=%lx, guid: %s
442: level=4,    text=VQ %p Target "%s"(%p): connected to state %s %s
443: level=3,    text=VQ %p target (name=%s, location=%s, type=%s) synchronizing
444: level=0,    text=mission impossible 2: Target %s@%s.%s (%d)
445: level=3,    text=uqm event id=%d(%s), type=%d, partition=%s, handle=%I64d received
446: level=3,    text=uqm event id=%d(%s), type=%d, partition=%s, handle=%s received
447: level=3,    text=uqm event answer(%s) on request type=%d(ref_id=%ld), handle=%s received
448: level=2,    text=assigning interaction to AP "%s" from partition "%s" (%s:%s, ref_id=%d)
449: level=3,    text=attach user data to interaction (ref_id=%d)
450: level=3,    text=update routing criteria: BS="%s", MS="%s", AP="%s", EAP="%s" (ref_id=%d)
451: level=4,    text=tenant %s, AP %s has %s = %s
452: level=4,    text=tenant %s, AP %s property %s %s, new value = %s
453: level=3,    text=Unhandled request is deleted
454: level=1,    text=Exception is received for HTTP Bridge %s
455: level=1,    text>CreateProcess failed. Error code: %d
456: level=1,    text=Fork failed! Errno=%d. %s
457: level=1,    text=Exec '%s' failed! Errno=%d. %s
458: level=1,    text=Process termination failed. Error code: %d
459: level=5,    text=PID file wasn't deleted after stopping HTTP Bridge
460: level=4,    text=HTTP Bridge started (pid=%ld)
461: level=1,    text=Can't start HTTP Bridge: %s
462: level=4,    text=Process stopped (pid=%ld)
463: level=3,    text=Close Process Handle failed. Error code: %u
464: level=3,    text=Close Thread Handle failed. Error code: %u
465: level=5,    text=Signal SIGKILL sent to process (pid=%ld)

```

```

466: level=4,    text=OK InfoMessage is received from server %s
467: level=1,    text=Error in message from HTTP Bridge: %s
468: level=4,    text=HTTP Bridge is exiting
469: level=1,    text=Unknown message ID received: %d
470: level=1,    text=Error sending message to HTTP Bridge
471: level=3,    text=SOAP request %ld sent to HTTP Bridge:
    URL:        %s
    Method:      %s
    NameSpace:   %s
    SOAPAction:  %s
    Input:       %s
    Output:      %s
    HTTPAuthent:%d %s %s
    SOAPSecrty: %s
472: level=0,    text=%s
473: level=0,    text=%s
474: level=0,    text=%s
475: level=0,    text=%s
476: level=0,    text=%s
477: level=0,    text=%s
478: level=0,    text=%s
479: level=0,    text=%s
480: level=0,    text=%s
481: level=0,    text=%s
482: level=0,    text=%s
483: level=0,    text=%s
484: level=0,    text=%s
485: level=0,    text=%s
486: level=0,    text=%s
487: level=0,    text=%s
488: level=0,    text=%s
489: level=0,    text=%s
490: level=0,    text=%s
491: level=0,    text=%s
492: level=0,    text=%s
493: level=3,    text=WFM schedule updated. %d records. StartTime: %s (%ld)
494: level=3,    text=WFM activities updated. %d activities total.
495: level=1,    text=WFM error! Agent %d not found in configuration server.
496: level=1,    text=out of resources: %s %s %
497: level=3,    text=VQ %p Target "%s"(%p) deleted
498: level=3,    text=binding with Target %p (VQ %p)
499: level=3,    text=Licensing seat %s has %d links: %c object <%s>, type <%s> (used seats=%d)
500: level=0,    text=mission impossible 4: object <%s>, type <%s>, block %s
501: level=5,    text=targets
502: level=5,    text=content
503: level=5,    text=calls
504: level=5,    text=targetinfo
505: level=2,    text=assign
506: level=1,    text=EventPrimaryChanged is received for tserver %s
507: level=4,    text=ready MEDIA %s[%d] (type=%d time=%ld) for agent %s(%ld), place %s(%ld), status %s
    time=%ld

```

```

508: level=3,    text=tenant %s activate content updating for <%s:%s>, #%ld->%ld %s, on timer %lu
509: level=1,    text=unknown function %s
510: level=3,    text=remote party changed detected for tserver %s (this dn=%s): %s -> %s
511: level=1,    text=configserver %s session %d is restored from socket %d
512: level=3,    text=object unsupported event %d: statistic <%s> object <%s>
513: level=5,    text=RStatCallsInQueue update: object <%s>(%ld), type %s value %d+%d+%d, reason <%s>
514: level=1,    text=EventUserEvent is received for tserver %s (this dn=%s)
515: level=0,    text=updating chain timer proc: %ld %ld %ld %p
516: level=2,    text=function will be continued(%d,%lu)
517: level=2,    text=function is continued(%d,%lu)
518: level=3,    text=pulse for one call
519: level=3,    text=change connid to %s
520: level=3,    text=connid %s is bound to the call %ld-%p
521: level=1,    text=MyEventMergeCalls is received for tserver %s (this dn=%s): %s -> %s
522: level=3,    text=there is no object for event %d from stat server %s
523: level=3,    text=select target by quota %d: %s
524: level=3,    text=select target by threshold %s %c %d : %s
525: level=4,    text=DN type %d is not active
526: level=4,    text=empty event (%d, %d, %ld)
527: level=1,    text=unknown message %s received from %d (%s '%s')
528: level=4,    text=VQ %p first ready call is %s (%d%d%d%d)
529: level=4,    text=VQ %p Target "%s"(%p) Component #%d %s: first ready call is %s
530: level=3,    text=VQ %p first available call: %s, reason=%s
531: level=1,    text=repeating of the last function
532: level=1,    text=send to tserver %s Cancelation of PrivateService %d on dn %s
533: level=3,    text=find %s statvalue of %s for targets: %s
534: level=5,    text=PID file wasn't deleted after stopping HTTP Interface
535: level=4,    text=HTTP Interface started (pid=%ld)
536: level=1,    text=can't start HTTP Interface: %s
537: level=3,    text=connid %s generated for client=%d(%s), ref id=%d, method name=%s
538: level=2,    text=routing interface %s request received: %s, client=%d(%s), ref=%d
539: level=1,    text=can not find tserver for event %s, tenant %s
540: level=5,    text=VQ %p Target "%s"(%p) check threshold: %s = %d
541: level=3,    text=STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: statistic <%s> renamed
542: level=4,    text=statistic ATT for MEDIA %s (type=%d), time=%ld: %s %d=%d+%d
543: level=2,    text=optimized routing iteration %d for rule %s starts
544: level=2,    text=optimized routing iteration %d for rule %s ends (%d calls dispatched)
545: level=4,    text=optimized routing distribution #%d: VQ(%p) target %p (%p)
546: level=3,    text=VQ %p adjusting: transaction %s, event %s
547: level=4,    text=tserver %s cdn %s added/updated: %sREG %s %s %s %s
548: level=4,    text=tserver %s cdn %s deleted
549: level=3,    text=target selection tuning: cost=%s, ladv=%s
550: level=2,    text=current call classification: media=%s(%lu), service=%s(%lu), segment=%s(%lu)
551: level=3,    text=update volume contracts tables: target %s, group %s, tenant %s
552: level=4,    text=update contract %s table %s zone %s day %s(%d-%d-%d) interval #%d(%d):
      [%d(%d), %d, %d, %d]->%ld(%ld) %d(%d)
553: level=4,    text=COST for %s@%s.%s (p0=%p, Target "%s"(%p)) asked: from %s to %s, contract %s
554: level=3,    text=cost from %s to %s, contract %s[table %s(%s,%s:%d)] returns %d+%d
555: level=3,    text=cost from %s to %s, contract %s[table %s(%s), zone %s, day %s(%d-%d-%d), interval
      #%d(%d, %d, %d, %d - %d)] returns %d+%d
556: level=1,    text=try %c-connecting: %s%s%s

```

557: level=3, text=current activity for agent %s (tenant %s): %s
558: level=3, text=call %s access resource %s (dbid=%ld)
559: level=3, text=router %ld %s access resource %s (dbid=%ld)
560: level=4, text=ready MEDIADN for %s[%d] (switch %s, number %s, type %d, time=%ld) for agent %s,
place %s, status %s time= %ld
561: level=3, text=list of access resources from switch %s to switch %s (%s) updated: %d entries, %d
available
562: level=5, text=access resources
563: level=4, text=number of available access resources from switch %s to switch %s (%s): %d (%c%d)
564: level=1, text=external routing application %s(%lu) %sabled
565: level=1, text=lost of messagesd from external routing application %s(%lu) detected:
expected=%d, received=%d
566: level=4, text=external router %lu reports calls in queue for %s %s: %d+%d
567: level=6, text=RStatCallsInQueue external details
568: level=4, text=external router %lu reports blocking of target: tenant %s agent %s place %s for %d
sec
569: level=3, text=HERE IS RDATA
570: level=3, text=request %ld to routerserver %s sent: %s(%s)
571: level=4, text=OK RInfoMessage is received from server %s
572: level=2, text=error in RInfoMessage from server %s, call is in status CALL_STATUS_RDATA
573: level=3, text=unhandled request is deleted
574: level=1, text=Exception is received for router server %s
575: level=2, text=routing interface response %s sent to client=%d(%s), ref=%d
576: level=4, text=dmessage
577: level=6, text=C%s (nMsgID=%d, nRequestID=%d, nError=%d): %s
578: level=5, text=routing interface response %s received from server=%d(%s), ref=%d
579: level=1, text=DEPARTURE+TIMER delivery
580: level=3, text=STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: verification mode - %d sec
581: level=3, text=EventRemoteConnection% is received for tserver %s
582: level=5, text=VQ %p Target "%s"(%p) Component %s check threshold: %s = %d
583: level=6, text=Monitor %lu %d
584: level=5, text=RStatLB details: LB=% .12g ALI=%d AR=%d(%d in % .12g) CIQ=%d RCIQ=%d
AHT=% .12g(% .12g % .12g)
585: level=5, text=RStatExpLB details: LB=% .12g ALI=%d AR=%d CIQ=%d AHT=% .12g(% .12g % .12g) D=% .12g
IR=(% .12g % .12g) SEQN=(%d %d %d) ECIT=% .12g ABE=% .12g CIQE=% .12g
586: level=6, text=RStatNthLAA details: VQ %p Target "%s"(%p) time=%ld, n=%d total=%ld
587: level=6, text=Trgt # %d
588: level=6, text=Cpnt # %d
589: level=1, text=time base adjusted by %ld: %ld -> %ld
590: level=3, text=%s request %ld sent to HTTP Bridge:
URL: %s
Header: %s
Content: %s
HTTPAuthent:%d %s %s

591: level=6, text=spare memory: %d
592: level=1, text=IRL generation failure: %s(%s)
593: level=3, text=virtual queues allocation pattern: static=(%d %u %u) dynamic=(%u %d/1024)
594: level=1, text=send to tserver %s TNetworkSingleStepTransfer to dn %s on %s through %s
595: level=1, text=EventNetworkCallStatus(%s) is received for tserver %s (this dn=%s)
596: level=1, text=there is no strategy item for %s

```

597: level=1, text=configserver socket %d is restorable, previously %d
598: level=5, text=statistic value conversion: %.12g->%.12g (%.12g %d %.12g)
599: level=5, text=time in ready state conversion at %ld: %d(%d)->%d(%d) (%d %.12g)
600: level=1, text=time shift with server %s is %d (%ld - %ld)
601: level=5, text=config update shunter = %ld
602: level=4, text=RVM started (id=%d)
603: level=1, text=cannot start RVM: %s
604: level=5, text=rvmmessage
605: level=0, text=mission impossible 5: timer %lu, call %p
606: level=2, text=function won't be continued(%d,%lu)
607: level=5, text=STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: %s content updating timer (%ld)
    for %d msec set
608: level=5, text=entering state <%s>
609: level=5, text=exiting state <%s>
610: level=1, text=simple request created: client=%s, ref=%d
611: level=1, text=simple request deleted: client=%s, ref=%d
612: level=2, text=routing failed%: %s
613: level=3, text=function %s operand %d: invalid type - %d
614: level=3, text=operand value out of range
615: level=3, text=object has no property "%s"
616: level=3, text=object property can not refer on object itself: "%s"
617: level=5, text=init SCXML session (%s)
618: level=5, text=%s SCXML session (%s)
619: level=5, text=%s SCXML look for transitions (%d)
620: level=5, text=%s SCXML check transitions for state %s, event %s
621: level=5, text=%s SCXML microstep %s
622: level=0, text=mission impossible 6: server %s, request id %lu
623: level=5, text=SCXML %s queue: event "%s" added
624: level=3, text=HERE IS INVOKE: state=%s, type=%s, src=%s
625: level=1, text=SCXML session with ID=%s wasn't found
626: level=5, text=SCXML transition condition evaluated to %d
627: level=6, text=MEMORY: object %p %s (%d)
628: level=3, text=read file '%s' from %d, bytes %d
629: level=1, text=error in message from async server: %s
630: level=4, text= async task completed by server %s
631: level=1, text= unknown RVM command %c
632: level=2, text=%s chance exception detected (function %s)
633: level=3, text= call's priority range from %d to %d
634: level=6, text= %s: current frame %s(%d)
635: level=2, text= target unreachable: %s
636: level=5, text=cancel active requests to tserver %
637: level=1, text= fail to parse string '%s' by symbols: %d %d
638: level=1, text= call partition set to %d, my cluster is %lu
639: level=6, text= call tpoints list:
640: level=5, text= server <%s> %d statistics reopened
641: level=4, text= STATOBJECT(%p %ld %d) tenant %s name=%s@%s.%s: first ready call is %s (VQ=%p)
642: level=3, text= pulse for one objct
643: level=3, text= check priority
644: level=6, text= STATOBJECT(%p %ld %d) tenant %s name=?%s:%s@%s.%s: update history %s
645: level=5, text= STATOBJECT(%p %ld %d) tenant=%s name=%s@%s.%s: statistic <%s> opening error: %s
646: level=4, text= STATOBJECT(%p %ld %d) tenant %s name=%s@%s.%s: %s media status: %d %d %d

```

```

647: level=1, text= bad client %s: %
648: level=1, text= WARNING: Attention! Interaction %s will be terminated, reason: %s
649: level=3, text= claim OCS %s for <%s> agents from %s(%s)
650: level=1, text= script %s cannot be called - %
651: level=1, text= client on socket %d(%d) registered: type=%d name="%s"
652: level=5, text= VQ %p Target "%s"(%p) Component #%d %s: mismatch=%d, %
653: level=5, text= VQ %p Target "%s"(%p) mismatch=%d, %
654: level=5 text= SCXML: session SID=%s initialized
655: level=3 text= S0(%p %ld %d) ten=%s name=%s@%s.%s: peek stat <%s> %ld (skt %ld)
656: level=3 text= sending event %d for vq %
657: level=5 text= call %s not found, message is stored (up to %d msec)
658: level=3 text= HERE IS WAIT EXTERNAL_EVENT(%d sec)
659: level=3 text= HERE IS WAIT ALL(%s) for %d sec
660: level=5 text= S0(%p %ld %d) ten=%s name=%s@%s.%s: content updating chain #%ld
completed
661: level=1 text= send to ts %s TRequestPrivateService %d on dn %
662: level=3 text= queued delay: queue=%s(%c,%ld), order %ld, delay=%ld, max=%ld
663: level=3 text= queued delay over: queue=%s(%c,%ld)
664: level=4 text= map <%s> key <%s> updated with value <%s>, timeout %d by %lu
665: level=1 text= Warm Standby (backup) mode activated
666: level=1 text= Hot Standby (backup) mode activated
667: level=1 text= Warm Standby (Primary) mode activated
668: level=1 text= Hot Standby (Primary) mode activated
669: level=3 text= current locale is <%s> MB=%d
670: level=3 text= percentage object %p for <%s> asked: w=%d, v=%.12g
671: level=3 text= percentage objects <%s...> cleared
672: level=5 text= calluuid %s is bound to the call %ld-%p as %p
673: level=5 text= change current calluuid %p -> %p
674: level=1 text= code conversion %s (locale=%s) failed for '%s'
675: level=2 text= stat's contents update flag set to %d hint=%p %s %s
676: level=5 text= agent reservation for %s cleared
677: level=5 text= S0(%p %ld %d) ten=%s name=%s@%s.%s stat <%s>: %
678: level=5 text= stat subscription %d (ten %s, stat %s, target %s, interval %d)
deleted (%d entries left)
679: level=5 text= stat subscription %d check threshold: %s => %d
680: level=5 text= stat subscriptions for ten %s, stat %s, target %s, interval %d
answering (%d entries)
681: level=1 text= RequestPrivateService is received for ts %s (this dn=%s)
682: level=5 text= timestamp for agent custom state %s new state %d: %ld
683: level=3 text= checkup subroutine %
684: level=4 text= tag %s for %s (ten %s) %s (%d, %ld)
685: level=3 text= target subscription %
686: level=3 text= call (vq %p) changes pos for VQ %p (name="%s"), from (id=%d, pr=%d,
pos=%ld) to (id=%d, pr=%d, pos=%ld)
687: level=1 text= Attention! interaction% performed a lot of % actions
688: level=3 text= WFM agent groups updating
689: level=5 text= number of calls from sw %s to sw %s (%lu%d): %d(%d)
690: level=3 text= call (%ld-%p) restoring(%d %d %d)
691: level=3 text= main call in progress: states: s=%d d=%d t=%d r=%x
692: level=1 text= call (%ld-%p) for %s passed to immortality
693: level=2 text= channel from sw %s to sw %s %

```

694 level=5 text= stop trying to route to ag "%s" (pl "%s"): %s
695 level=4 text= call index <%s> key <%s> %s



Appendix

B

IRD Web Service Object

Note: The IRD Web Service strategy building object can be used in both voice and non-voice routing strategies.

URS provides a tool, *HTTP Bridge*, for communicating with Web Services through SOAP/XML over HTTP/HTTPS protocols. Previously, in 7.0.1, URS used HTTP Bridge to extract agent scheduling information from Genesys Workforce Management. Starting with 7.2, HTTP Bridge was extended to allow strategy developers to communicate with Web Services applications outside of Genesys via the Web Service IRD strategy-building object.

This Appendix includes the following sections:

- [URS Options Affecting Web Service, page 772](#)
- [Web Service Object, page 772](#)
- [General Tab, page 774](#)
- [Security Tab, page 778](#)
- [Result Tab, page 786](#)
- [How the Web Service Object Works, page 790](#)
- [Another Web Service Example, page 791](#)
- [Web Services Resources, page 796](#)

URS Options Affecting Web Service

Before using the Web Service object, you must configure the various options in the URS Application object. There are General, Log, SOAP, and WFM7 HTTP Bridge options (see Web Services/Web API Options, [page 687](#)) that control tracing and debugging when you use HTTP Bridge to access Web Services and to adjust HTTP Bridge performance characteristics. For information on setting on these options, see the section on using Web Services in the “Manually Configuring Applications” chapter in the *Universal Routing 8.1 Deployment Guide*.

Web Service Object

Access the Web Service object in the Routing Design window from the Data & Services toolbar (see [Figure 261](#)).

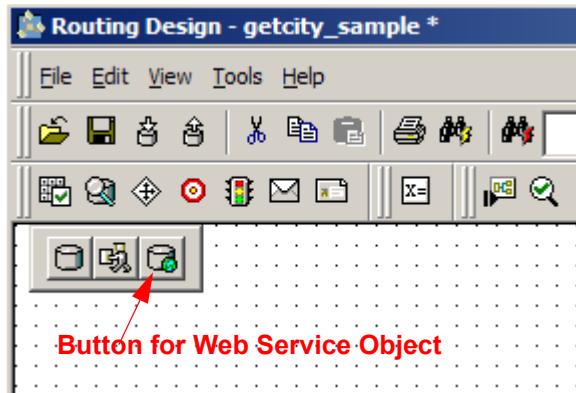


Figure 261: Button for Web Service Object

Use the Web Service object if you are using the Gplus Adapter for mySAP Data Access Component. Or use it to access a Web Service on the Internet (see “Another Web Service Example” on [page 791](#)).

A Web Service object specifies SOAP parameters (including the method name) and request parameters in the form of key-value pairs, which include the zip code to be converted into a city name. SOAP is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS.

Sample Strategy

[Figure 262](#) shows a strategy called `GETCITY_SAMPLE` that uses the Web Service object.

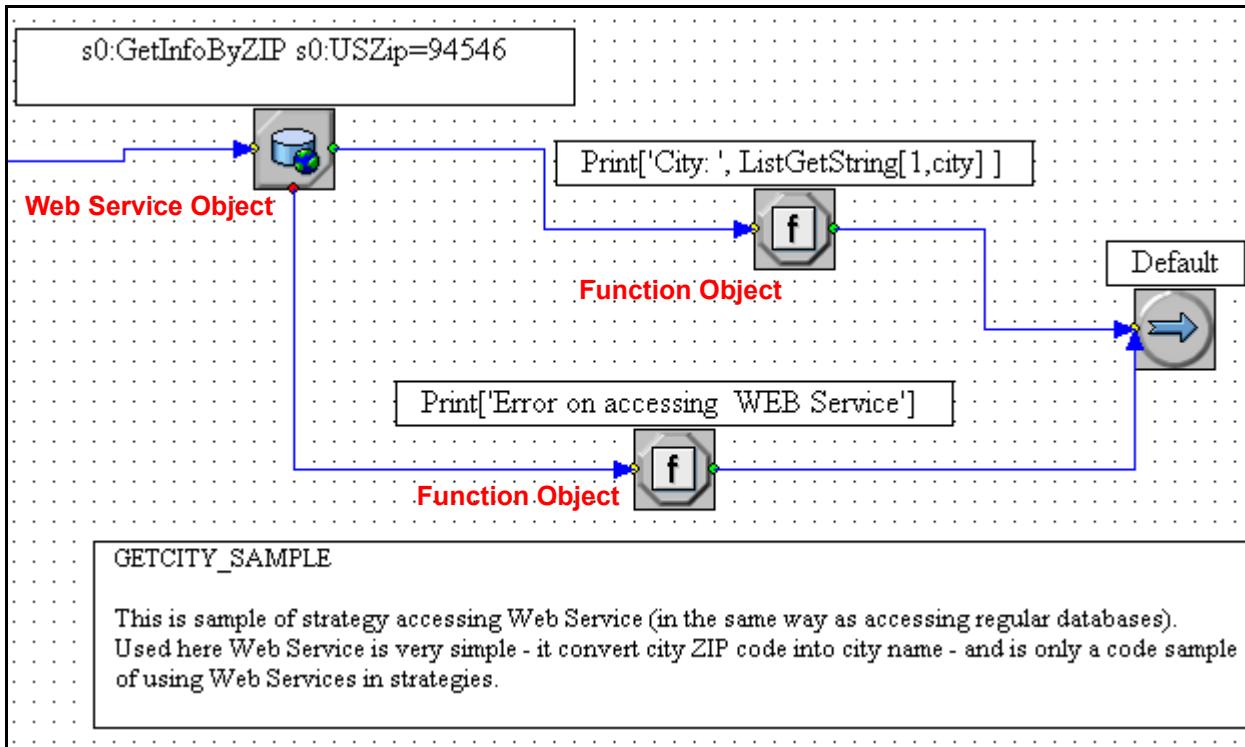


Figure 262: GETCITY_SAMPLE Strategy

This example accesses a Web Service that converts a city zip into a city name. The flow of `GETCITY_SAMPLE` shown in [Figure 262](#) is discussed below.

Note: When you install IRD, you automatically get the strategy bytecode for `GETCITY_SAMPLE`. The sample shows how accessing a Web Service from a strategy is very similar to accessing data in a database from a strategy. The purpose of `GETCITY_SAMPLE` is simple: it converts a city zip code into a city name.

General Tab

Starting with 8.1.2, the Web Service object provides the option to select SOAP or plain HTTP requests.

Figure 263 shows the General tab of the Web Service object for a SOAP request shown in Figure 262.

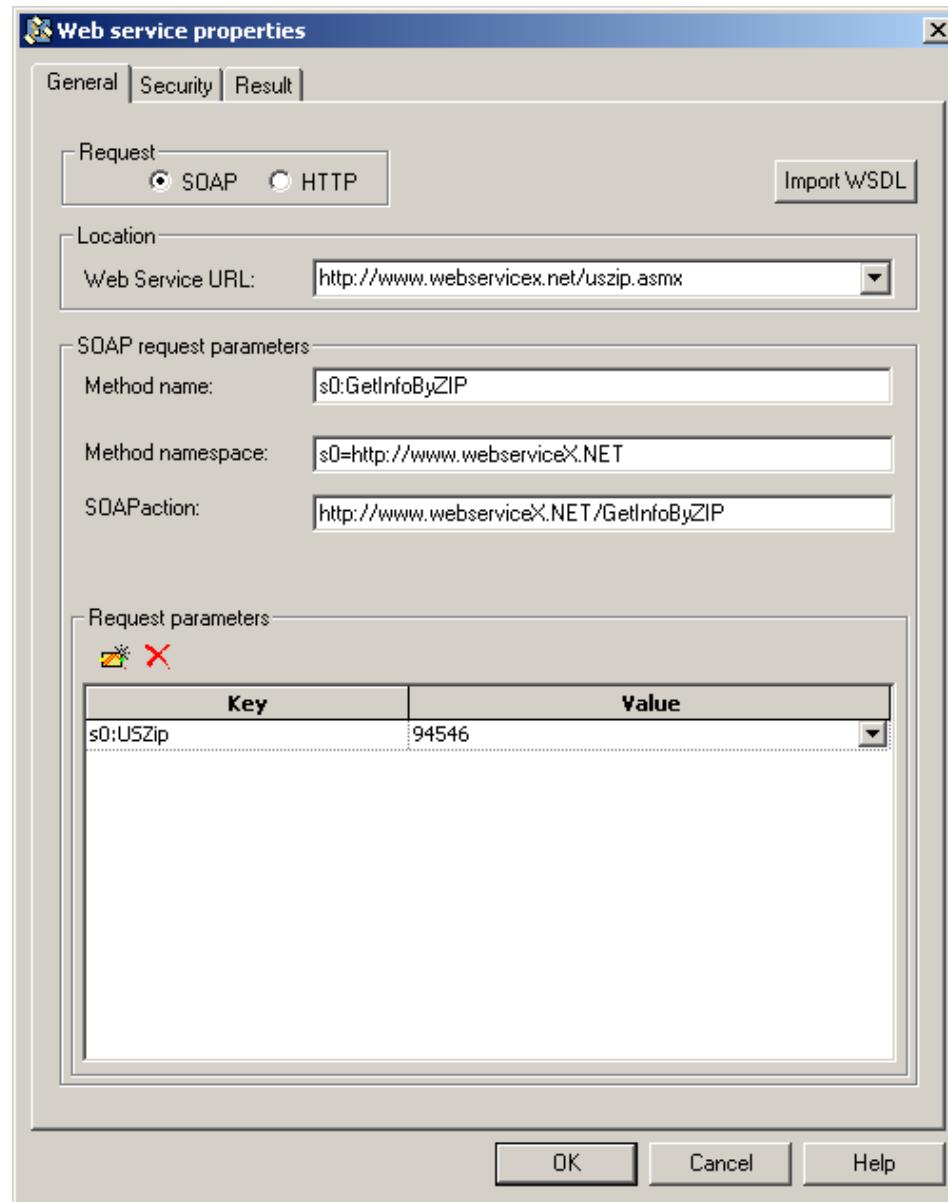


Figure 263: Web Service Object General Tab for SOAP request

To configure the Web Service object's General tab:

1. Enter the URL for the Web Service you want to access (mandatory).
 - In [Figure 263](#), the entry is: `http://www.webservicex.net/uszip.asmx`
 - Another example:
`http://services.xmethods.net:80/soap/servlet/rpcrouter`
2. Enter the remote procedure (method) name (mandatory).
 - In [Figure 263](#), the entry is: `s0:GetInfoByZIP`
 - Another example: `getTemp`
3. Enter the XML namespace, which is a collection of names having the general form "`ns1=http://example.com/...`". If you specify multiple namespaces, they should be separated by commas.

A namespace represents an application-dependent or context-dependent URI that is used in XML documents as an element type and attribute name. Usually the namespace is mandatory, but in some cases it may be empty, depending on the Web Service. If empty, only default namespaces are used.

 - In [Figure 263](#), the entry is: `s0=http://www.webserviceX.NET`
 - Another example: `ns=urn:xmethods-Temperature`
4. In the `SOAPaction` text box, you can enter optional HTTP header data.
 - In [Figure 263](#), the entry is: `http://www.webserviceX.NET/GetInfoByZIP`
5. Enter the request parameters in the form of key-value pairs. These are the parameters for the procedure (method).

These function parameter values can be obtained from the Web Service description on the Web Service provider's website or from the WSDL description file.

6. Entering parameters is usually mandatory, but in some cases may not be necessary, depending on the Web Service.
 - In [Figure 263](#), the entry is: `Key=s0:USZip, value=94546`
 - Another example: `Key=acctnum, value=5390021`

[Figure 264](#) shows the General tab of the Web Service object for an HTTP request.

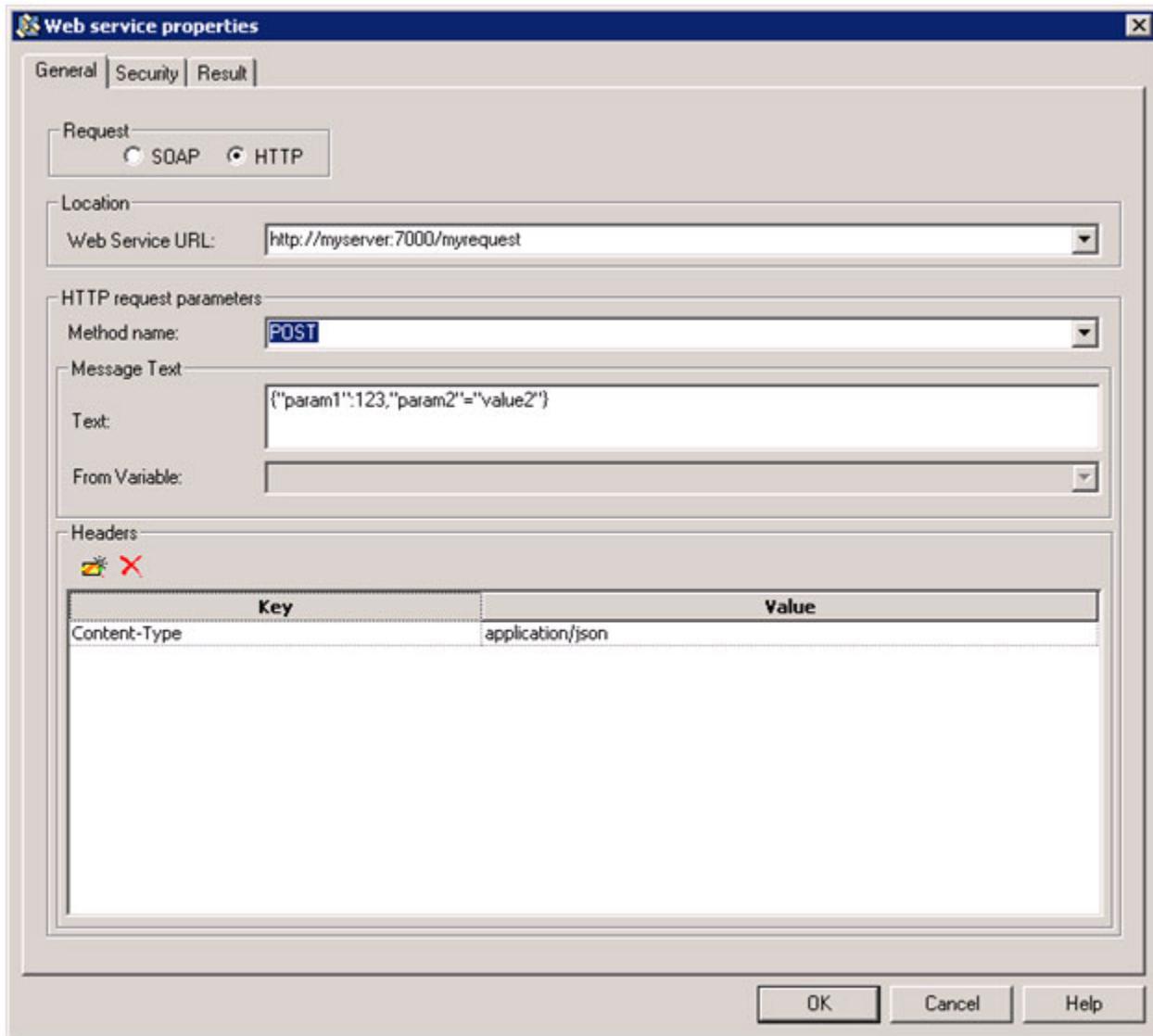


Figure 264: Web Service Object General Tab for HTTP request

To configure the Web Service object's General tab:

1. In the Web Service URL text box, enter the URL for the Web service you want to access (mandatory).
 - In [Figure 264](#), the entry is: `http://myserver:7000/myrequest`
2. In the Method name text box, select from the pull down menu the HTTP request method to indicate the desired action to be performed on the identified resource (parameter #1 URL) (mandatory).
 - In [Figure 264](#), the entry is: POST
3. In the Text box, you can directly provide a message body of the HTTP request (optional).

- In [Figure 264](#), the entry is: {"parm1":123, "parm2"="value2"}
4. In the From variable text box, you can enter a message body of the HTTP request provided as a value of the selected variable (optional).
 5. Set of HTTP Headers (list of key value pairs) that can be added to the HTTP requests (optional).
For example:
In the case of a non empty message body, this can be used to inform the server of the format of the body content through a Content-Type header.
 - In [Figure 264](#), the entry is:
Key=Content-Type, value=application/json

For more details on HTTP protocols, please reference to the following:

http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Nested WSDL Access

The IRD Web Service object can process names of *nested* parameters using a “dotted” format such as Key1.Key2 for example. In this way you can manually enter nested input parameters.

If the Web Service object has just one parameter and its name is Body, then its value will be interpreted as the entire SOAP request. This allows you to manually create and use the XML text of SOAP requests.

For example, assume a Web Service method has *two* input parameters, param1 and param2. If an object (XML element) with properties Key1, Key2, and Key3 should be passed as the value of param1, with param2 being just any string, the input parameters may appear as shown in [Figure 265](#) on [page 778](#).

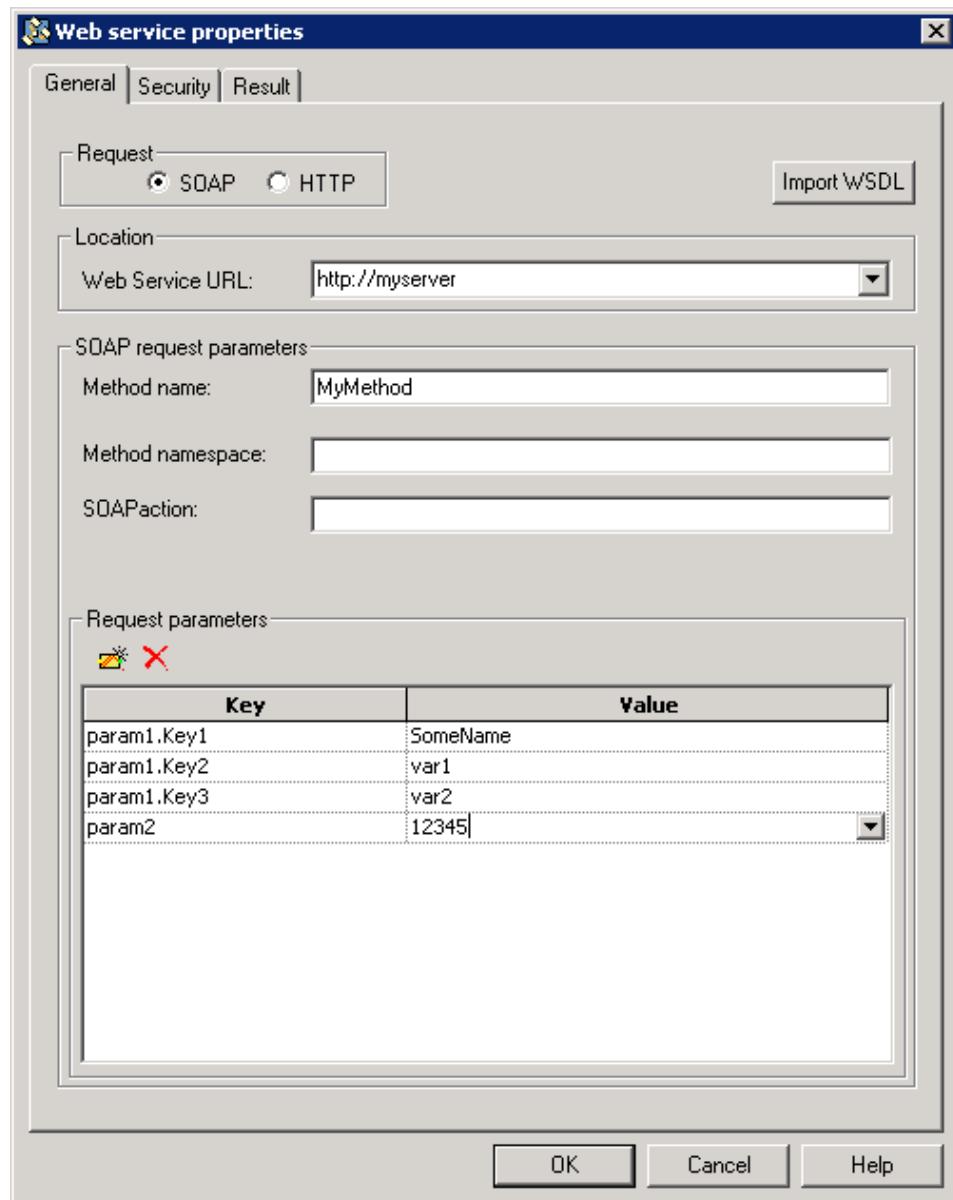


Figure 265: Nested WSDL Access

Security Tab

The Security tab in the Web Service properties dialog allows you to specify HTTP authentication and SOAP security settings that HTTP Bridge will use when invoking a remote Web Service. [Figure 266](#) shows the Security tab in the Web Service object filled in with sample data.

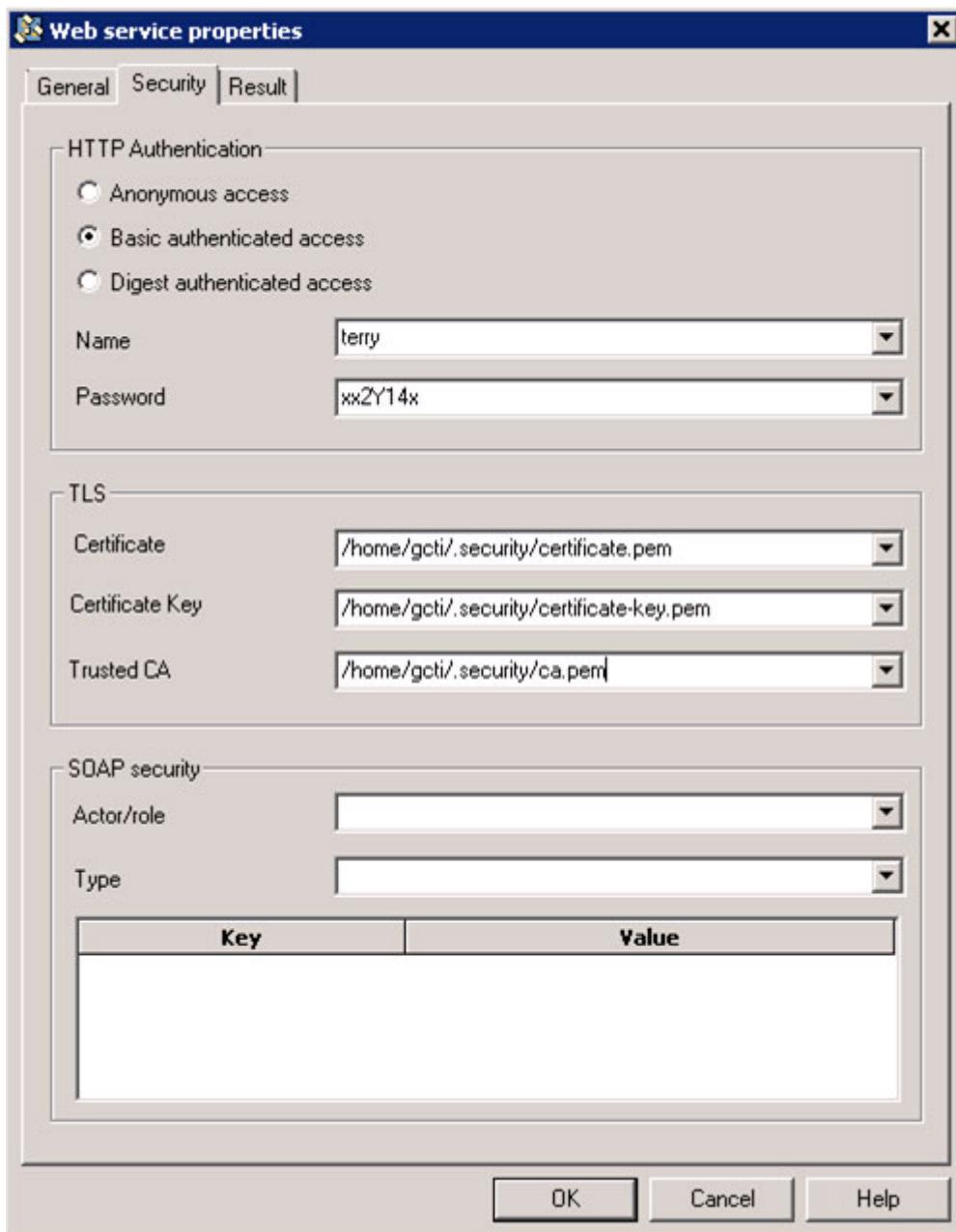


Figure 266: Web Service Object, Security Tab

HTTP Authentication Area

The HTTP Authentication section of the Security tab shown in Figure 266 on page 779 reflects HTTP protocol. It provides you with the following three access options:

1. Anonymous access (default)

With this type of access, no user name/password is passed to Web service for client authentication in order to get data.

2. Basic authenticated access (see “[Basic Authentication](#)” below)
3. Digest authenticated access (see “[Digest Authentication](#)” below)

HTTP protocol provides a simple challenge-response authentication mechanism that may be used by a server to challenge a client request and by a client to provide authentication information. The Security tab reflects these two forms of HTTP authenticated access: Basic and Digest.

If you select Basic or Digest, then content of fields Name and Password are passed to the Web Service. The Web Service can then reject or grant access to the requested data based on passed authentication data.

Note: For some Web Services, a user name and password are required so these fields must not be empty. For other Web Services, these fields can be empty. For still other Web Services it does not matter what authentication mode you use at all. The Web Service object reflects this by allowing you to leave both fields empty. An empty string is a valid Web Service object parameter even if remote Web Service requires it.

Warning! The Password field of the Web Service object can accept a variable and its content is unprotected. When providing a password (if needed) for the Web Service object, Genesys recommends that you obtain the correct password from a secure source, store it in a variable, and then use the value of this variable as the value for the Password field.

Basic Authentication

In the context of an HTTP transaction, the Basic access authentication is a method designed to allow a web browser, or other client program, to provide credentials—in the form of a user name and password—when making a request. Although the scheme is easily implemented, it relies on the assumption that the connection between the client and server computers is secure and can be trusted. Specifically, the credentials are passed as plain text and could be intercepted easily. The scheme also provides no protection for the information passed back from the server.

To prevent the user name and password being read directly by a person, credentials are encoded as a sequence of base-64 characters before transmission.

Here is a typical transaction between an HTTP client and an HTTP server running on the local machine (localhost). It comprises the following steps.

- The client asks for a page that requires authentication but does not provide a user name and password. Typically this is because the user simply entered the address or followed a link to the page.
- The server responds with a 401 (Unauthorized) response code and provides the authentication realm.
- At this point, the client will present the authentication realm (typically a description of the computer or system being accessed) to the user and prompt for a user name and password. The user may decide to cancel at this point.
- Once a user name and password have been supplied, the client re-sends the same request but includes the authentication header.
- In this example, the server accepts the authentication and the page is returned. If the user name is invalid or the password incorrect, the server might return 401 (Unauthorized) response code and the client would prompt the user again.

A client may preemptively send the authentication header in its first request.

HTTP Bridge will use this form of authentication when the Basic authenticated access radio button (see Figure 266 on [page 779](#)) is selected. In this case, Name and Password fields in the HTTP Authentication section must be entered.

With this form of authentication, HTTP Bridge will include authentication header in its first request to the Web Service, which reduces the number of interactions between HTTP Bridge and the Web Service that are needed to carry out the request.

Digest Authentication

This method builds upon the Basic access authentication, allowing user identity to be established without having to send a password in plaintext over the network.

This typical transaction consists of the following steps.

- The client asks for a page that requires authentication, but does not provide a user name and password. Typically, this is because the user simply entered the address or followed a link to the page.
- The server responds with a 401 (Unauthorized) response code, providing the authentication realm and a randomly-generated, single-use value called a *nonce*.
- At this point, the client will present the authentication realm (typically a description of the computer or system being accessed) to the user and prompt for a user name and password. The user may decide to cancel at this point.

- Once a user name and password have been supplied, the client re-sends the same request but adds an authentication header that includes the “response” code.
- In this example, the server accepts the authentication and the page is returned. If the user name is invalid and/or the password is incorrect, the server might return a 401 (Unauthorized) response code and the client would prompt the user again.

A client may already have the required user name and password without needing to prompt the user, in other words, if they have previously been stored by a web browser.

The “response” value is calculated in three steps, as follows. Where values are combined, they are delimited by colon symbols.

1. The MD5 hash of the combined user name, authentication realm and password is calculated. The result is referred to as HA1.
2. The MD5 hash of the combined HTTP method and URI is calculated. The result is referred to as HA2.
3. The MD5 hash of the combined HA1 result, server nonce (nonce), request counter (nc), client nonce (cnonce), quality of protection code (qop) and HA2 result is calculated. The result is the “response” value provided by the client.

Since the server has the same information as the client, the response can be checked by performing the same calculation.

HTTP Bridge uses this form of authentication when the Digest authenticated access radio button (see Figure 266 on [page 779](#)) is selected. In this case, you must also enter the Name and Password fields in the HTTP Authentication section.

Additional information

<http://tools.ietf.org/html/rfc2617>

http://en.wikipedia.org/wiki/Basic_access_authentication

http://en.wikipedia.org/wiki/Digest_access_authentication

TLS

The information in this section refers to the fields in the TLS area of the Security tab shown in Figure 266 on [page 779](#).

SSL/TLS data for HTTPS requests are provided in the URS application in the section web. Options are: def_certificate, def_certificate_key and def_trusted_ca.

This means that all HTTPS requests made by Router use the same set of TLS/SSL parameters.

Beginning in URS 8.1.4, to provide the possibility of specifying TLS/SSL security parameters individually per request, the security tab of Web Service access object in IRD is extended with three optional parameters:

- Certificate
- Certificate Key
- Trusted CA

The values for these fields can be provided as variables or entered directly on the Web Service Object, Security Tab.

If a Web Service object has TLS parameters specified, the corresponding web request will use them.

If the Web Service object has no TLS parameters specified, the corresponding web request will take values from the URS application object, section web.

In either case, if the TLS parameters which are provided are incorrect, such as syntactically invalid, or referring to an expired certificate, this results in a failure to execute the given web request. The strategy execution continues through the red port of the web service object with a "0013 Remote Error" signaled.

Additional Information

For information on the meaning of TLS/SSL parameters certificate, certificate key, and trusted CA, see the section "Web Service Connections Using HTTP Bridge" in the *Genesys 8.1 Universal Routing Deployment Guide*.

Soap Security Area

The information in this section refers to the fields in the Soap Security area of the Security tab shown in Figure 266 on [page 779](#).

The Web Services Security specification (WS-Security) (<http://www.cgisecurity.com/ws/ws-secure.pdf>) provides a set of mechanisms to help secure SOAP message exchanges between Web Services. Specifically, WS-Security describes enhancements to the existing SOAP messaging to provide quality of protection through the application of message integrity, message confidentiality, and single message authentication to SOAP messages. These basic mechanisms can be combined in various ways to accommodate building a wide variety of security models using a variety of cryptographic technologies.

Additionally, WS-Security describes how to encode binary security tokens and attach them to SOAP messages. Specifically, the WS-Security profile specifications describes how to encode Username Tokens, X.509 Tokens, SAML Tokens, REL Tokens and Kerberos Tokens as well as how to include opaque encrypted keys as a sample of different binary token types. With WS-Security, the domain of these mechanisms can be extended by carrying authentication information in Web Service requests. WS-Security also includes

extensibility mechanisms that can be used to further describe the credentials that are included with a message. WS-Security is a building block that can be used in conjunction with other Web Service protocols to address a wide variety of application security requirements.

Message integrity is provided by leveraging XML Signature and security tokens to ensure that messages have originated from the appropriate sender and were not modified in transit. Similarly, message confidentiality leverages XML Encryption and security tokens to keep portions of a SOAP message confidential.

You can specify the following options in the `SOAP security` section of the Security tab (see Figure 266 on [page 779](#)):

- `Actor/role`
- `Type`
 - `UsernameTokenDigest`
 - `UsernameTokenText`
 - `BinarySecurityTokenPEM`

Actor/role

The `<Security>` header block of a SOAP message provides a mechanism for attaching security-related information targeted at a specific receiver (SOAP actor). This may be either the ultimate receiver of the message or an intermediary.

Use the `Actor/role` field to specify the target receiver for the `<Security>` header block. If this field is empty, the `<Security>` header block can be consumed by anyone.

Security Token Elements

Elements are added to the `<Security>` header block. As they are added, they are prepended to the existing elements.

UsernameToken Element

The `<UsernameToken>` element is a way of proving a username and optional password information.

The following illustrates the syntax of this element:

```
<UsernameToken Id="...">
<Username>...</Username>
<Password Type="...">...</Password>
</UsernameToken>
```

The `/UsernameToken/Password/@Type` attribute can have the following values:

- `wsse:PasswordText`

The value of the `/UsernameToken/Password` element is the actual password for the username. To use this option, select `UsernameTokenText` in the Type field in the SOAP security area (see Figure 266 on [page 779](#)) and specify values for the Name and Password keys.

- `wsse:PasswordDigest`

The value of the `/UsernameToken/Password` element is the digest of the password for the username. The value is a base64-encoded SHA1 hash value of the UTF8-encoded password. To use this option, select `UsernameTokenDigest` in the Type field in the SOAP security area (see Figure 266 on [page 779](#)) and specify values for the Name and Password keys.

BinarySecurityToken Element

The `<BinarySecurityToken>` element defines a security token that is binary encoded. The encoding is specified using the `EncodingType` attribute, and the value type and space are specified using the `ValueType` attribute.

This element has the following syntax:

```
<BinarySecurityToken Id=...  
EncodingType=...  
ValueType=.../>
```

To attach a binary security token stored in a PEM file to a Security header block, select `BinarySecurityTokenPEM` from the Type field in the SOAP security area and specify the name of the PEM file as the value for the Name key in Figure 266 on [page 779](#). Use a valid filename so that HTTP Bridge can access the file at runtime.

Result Tab

You specify what data you expect to be returned and what to do with it on the Result tab. [Figure 267](#) shows the Result tab of the Web Service object shown in [Figure 262](#) on page [773](#).

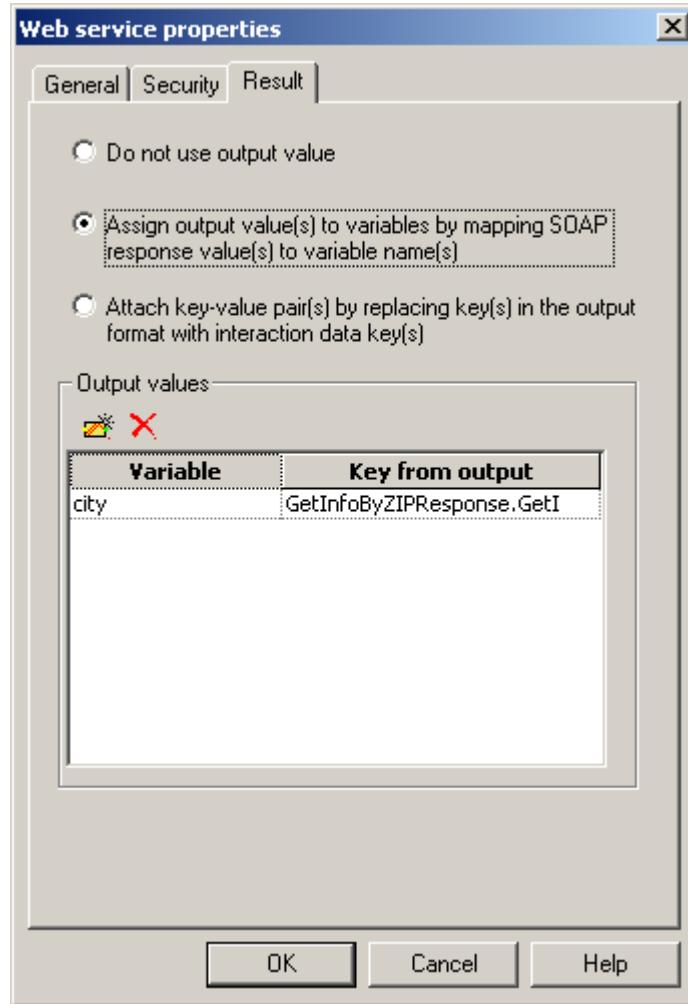


Figure 267: Web Service Object Result Tab

To configure the Result tab shown in [Figure 267](#):

1. Select the appropriate radio button to specify how to handle returned results.
2. If you select to assign the output values to variables, enter the variable names and the corresponding key from the return on the Output values pane.

The key must be the full path to the appropriate parameter(s) in the SOAP response.

- In [Figure 267](#), the entry is:
GetInfoByZIPResponse.GetInfoByZIPResult.NewDataSet.Table.CITY
- Another example: `getTempResponse.return`

Function Object #1

If the Web Service object does not return an error, the interaction goes out the side port to a Function object (the top Function object shown in Figure 262 on [page 773](#)). [Figure 268](#) shows the properties dialog box for this Function object.

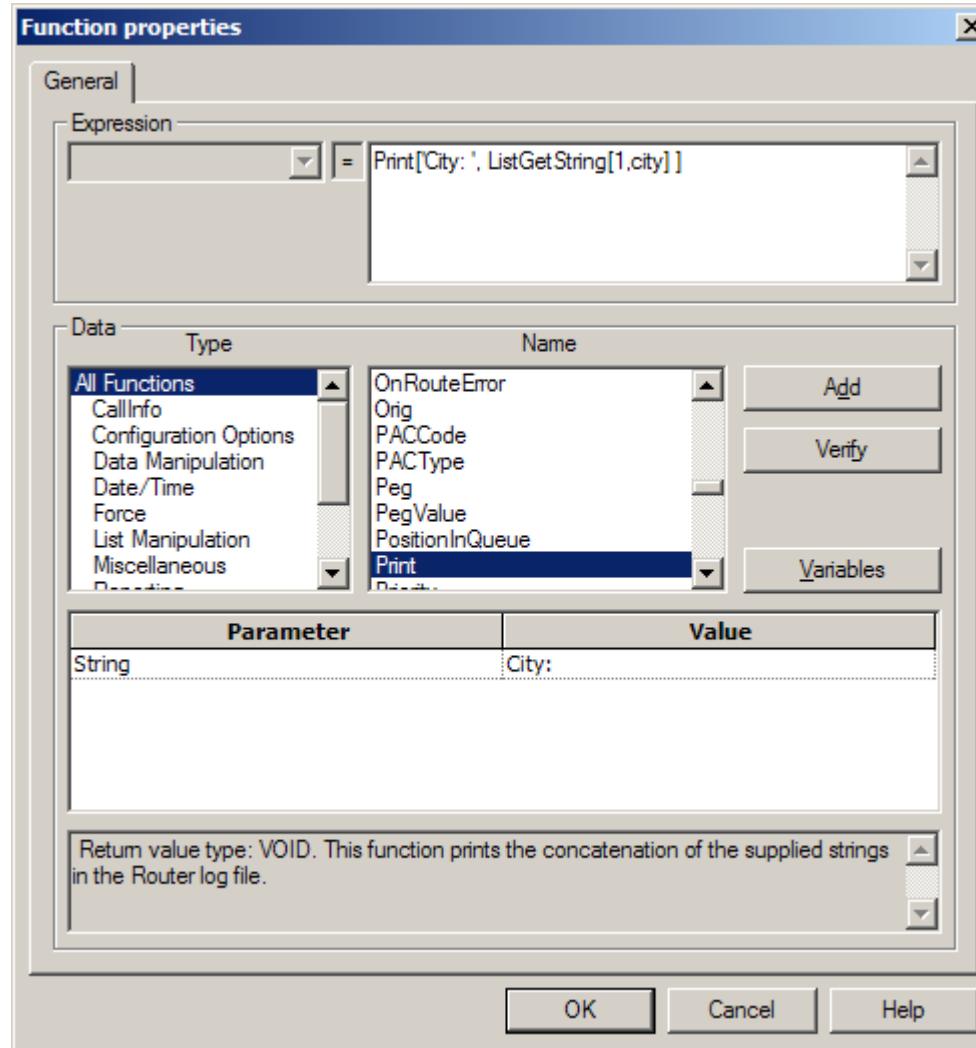


Figure 268: Function Object Specifying Print Function

The Function object uses two functions: the IRD Print and the IRD ListGetString Function.

- The Print function uses the output of ListGetString as a parameter.
- ListGetString takes the value of the `city` variable shown in [Figure 267](#) on [page 786](#).

Below are the function descriptions from the *Universal Routing 8.1 Reference Manual*.

ListGetString

Parameters: Index: INTEGER or variable

List: STRING (list) or variable (representing a string for a list)

Return value type: STRING

Parses out the string value for the specified key from a key-value list based on the position, starting from 1, rather than the key.

For example, if the list is key1:value1|key2:value2|key3:value3, then ListGetString(3, list) returns the value3 for key3. Any Index greater than the number of key-value pairs returns “ ” (an empty string), which means the Index is out of bounds.

Print

Parameter: String: STRING or variable (representing a string). The total length should not exceed 16,000 bytes.

Return value type: VOID

This function prints strings in the URS log file. Use a variable if you want to print a series of strings. The Print function in Universal Routing 7.x prints the string in the log at the interaction-level. This means it can show up in the Solution Control Interface (SCI), which allows trouble shooting from SCI.

Note: You could also specify a different function in the Function object and assign the output from the Web Service to a variable.

Function Object #2

If the Web Service object returns an error, the interaction goes out the bottom port to another Function object (the bottom Function object shown in Figure 262 on [page 773](#)). [Figure 269](#) shows the properties dialog box for this Function object.

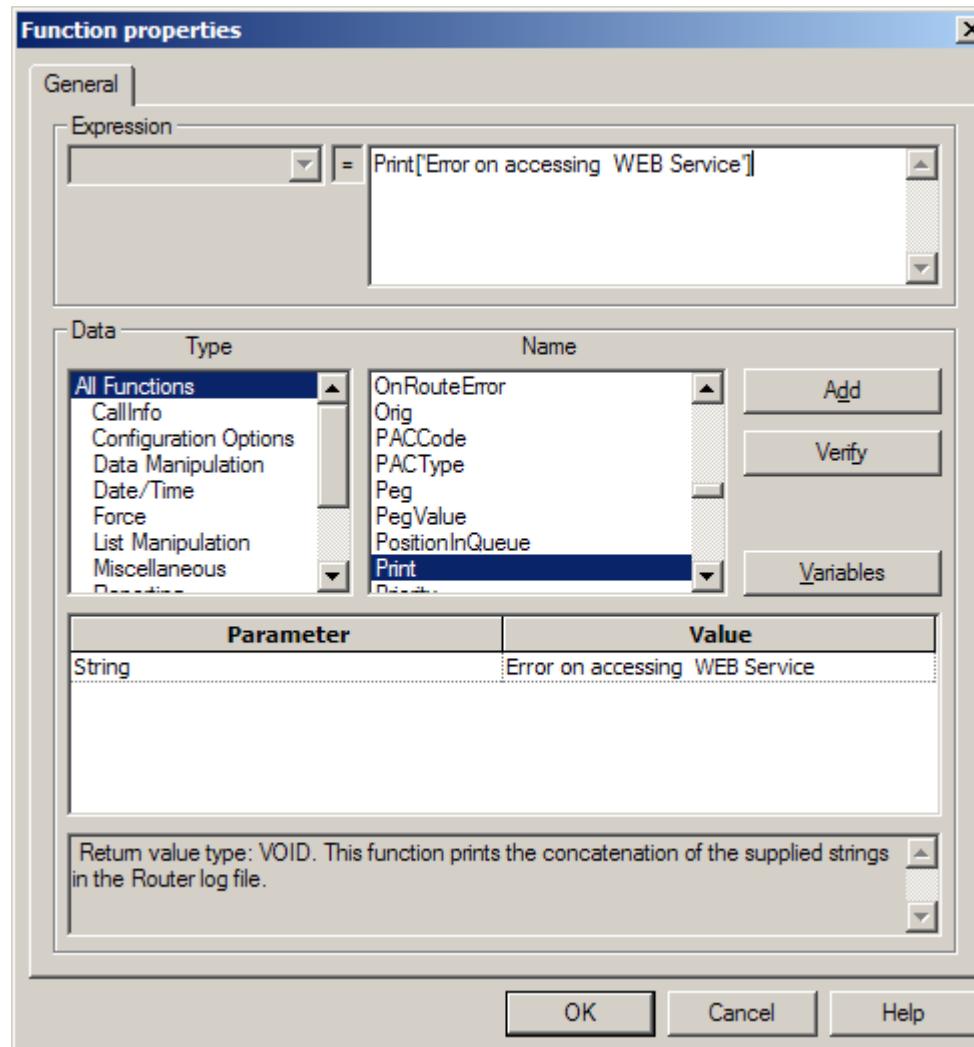


Figure 269: Function Object Used for Error Processing

Note: Don't be confused by the lack of other IRD objects in the GETCITY_SAMPLE strategy shown in Figure 262 on [page 773](#). The strategy is supplied only to show how to use the Web Service object. The strategy that you create using the Web Service object will contain other objects.

How the Web Service Object Works

When an interaction enters the Web Service object, URS:

1. Connects to the web server given in Web Service URL. If the URL starts with `https://`, URS uses a secured SSL connection.

Note: By default the remote connection port is **80** for HTTP and **443** for HTTPS.

2. Using given function parameters, constructs and sends a SOAP request over HTTP using the POST method.
3. Waits for the SOAP response from the remote Web Service.
4. Parses the SOAP response message and returns the output key values to the strategy that is handling the interaction.
5. If the communication with the remote Web Service is successful, the interaction proceeds through the Web Service object's green port.

In case of a failure (for example, a failure of the TCP/IP connection, a SSL handshake failure, or a timeout waiting for response), URS makes retry attempts.

After expiration of the maximum number of attempts specified in the `soap_retry_attempts` option, the interaction proceeds through the Web Service object's red port.

The URS installation installs the `GUR.wsdl` and `cfgschema.xsd` file.

The `GUR.wsdl` file is the description of the URS Web Service and relies on `gsd:list_pair` type definition provided by Genesys Configuration Server schema (`cfdschema.xsd` - a narrowed down version of Configuration Server schema is in the attached `.xsd` file). Also, the IRD installation installs the `ird-strategy-schema.xsd` file.

Both the `.wsdl` and `.xsd` file should be placed in the same directory to be used with an XML editor.

Limitation of the Web Service Object

The limitations are as follows:

- URS always expects that more than one element will be returned (=array). It packs the result into a key-value list string in the format `1:value|2:value....`. If the result contains only one value, it still has the format `1:value`. Use one of list manipulation functions to extract the value itself.
- No binary data is supported (a limitation of the strategy language).

- The Import WSDL functionality cannot parse all types (for example, it can't parse array types). You can enter the keys manually.
- WebService block does not allow to specify XML attributes for the method and input parameters of a SOAP request.

Another Web Service Example

The X-Methods Currency Exchange Rate Web Service returns the current exchange rate at which currency A is converted to currency B.

You can find the parameters in the Web Service description on the Web Service provider's Web site or from WSDL description file.

WSDL Description

Assume the X-Methods Currency Exchange Rate Web Service WSDL description file look like this:

```
http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl
<?xml version="1.0" ?>
<definitions name="CurrencyExchangeService"
  targetNamespace="http://www.xmethods.net/sd/CurrencyExchangeService
.wsdl"

  xmlns:tns="http://www.xmethods.net/sd/CurrencyExchangeService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="getRateRequest">
    <part name="country1" type="xsd:string" />
    <part name="country2" type="xsd:string" />
  </message>
  <message name="getRateResponse">
    <part name="Result" type="xsd:float" />
  </message>
  <portType name="CurrencyExchangePortType">
    <operation name="getRate">
      <input message="tns:getRateRequest" />
      <output message="tns:getRateResponse" />
    </operation>
  </portType>
  <binding name="CurrencyExchangeBinding"
    type="tns:CurrencyExchangePortType">
    <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"
    />
    <operation name="getRate">
```

```

        <soap:operation soapAction="" />
        <input>
            <soap:body use="encoded"
                namespace="urn:xmethods-CurrencyExchange"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            />
        </input>
        <output>
            <soap:body use="encoded"
                namespace="urn:xmethods-CurrencyExchange"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            />
        </output>
    </operation>
</binding>
<service name="CurrencyExchangeService">
    <documentation>Returns the exchange rate between the two currencies</documentation>
    <port name="CurrencyExchangePort"
        binding="tns:CurrencyExchangeBinding">
        <soap:address location="http://services.xmethods.net:80/soap"
    />
    </port>
</service>
</definitions>

```

General Tab in Web Services Object

We can get Web Services object parameters from the above WSDL description. Therefore, the Web Services object parameters for accessing the Currency Exchange Rate Web Service are the following:

- **Web Service URL:** `http://services.xmethods.net:80/soap`
- **Method name:** `ns:getRate`
- **Method namespace:** `ns=urn:xmethods-CurrencyExchange`
- **SOAP action:**
- **Request parameters:** `country1=UK, country2=USA`

Using the above entries, the General tab would appear as shown in [Figure 270](#):

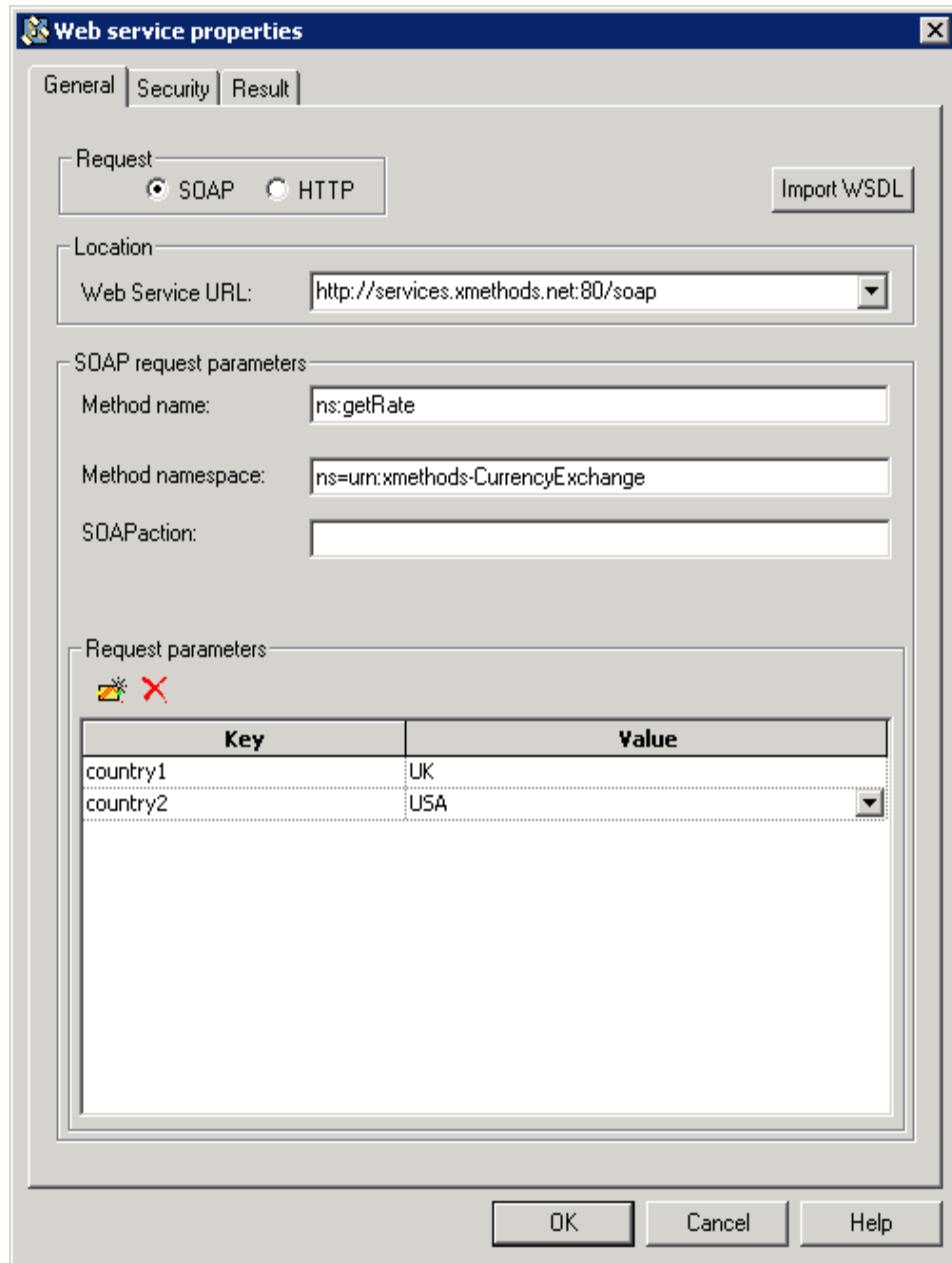


Figure 270: Web Services General Tab, Example 2

Result Tab in Web Services Object Properties

The WSDL description gives the following Web Services object parameters for the Result tab:

- **Output values:** getRateResponse.Result

Note: The float Web Service result type is ignored. The Web Services object always returns a string type result.

Using the above entry, the Result tab would appear as shown in [Figure 271](#):

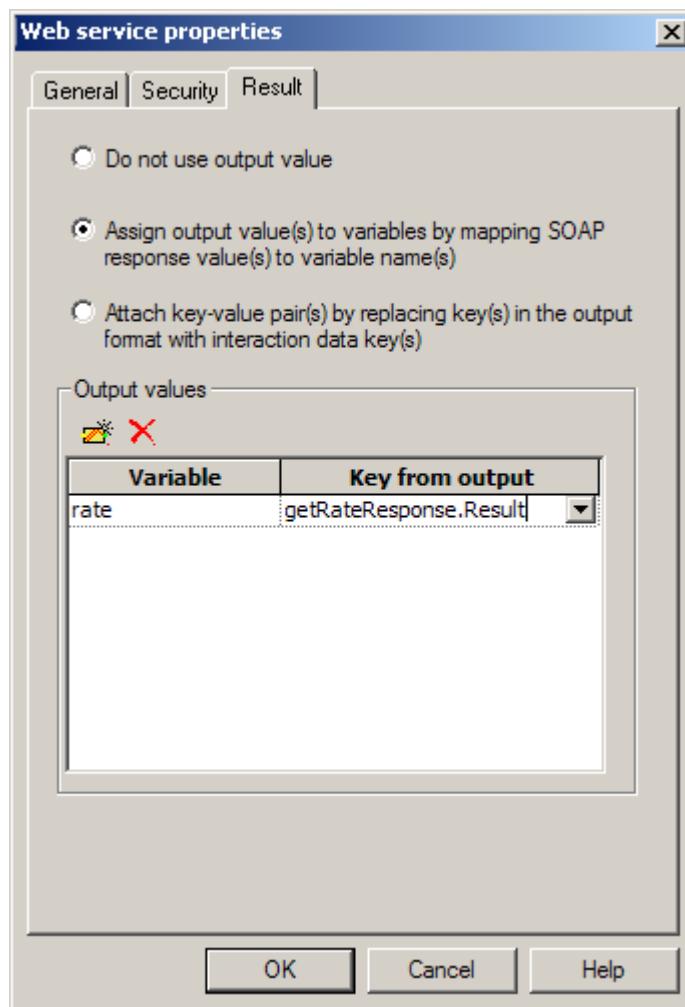


Figure 271: Web Services Result Tab, Example 2

URS SOAP Request Message

Using these parameters, URS sends the following SOAP message to the X-Methods Currency Exchange Rate Web Service:

```
POST /soap HTTP/1.0
Host: services.xmethods.net
User-Agent:
Content-Type: text/xml; charset=utf-8
Content-Length: 505
Connection: close
SOAPAction: ""

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
```

```

    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema"
    xmlns:ns="urn:xmethods-CurrencyExchange">
<SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<ns:getRate>
<country1>UK</country1>
<country2>USA</country2>
</ns:getRate>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Web Service Response Message

The X-Methods Currency Exchange Rate Web Service responds with this SOAP response message:

```

HTTP/1.1 200 OK
Date: Tue, 21 Jan 2003 02:05:24 GMT
Server: Electric/1.0
Content-Type: text/xml
Content-Length: 492
X-Cache: MISS from www.xmethods.net
Connection: close

<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope
  xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
  xmlns:xsi='http://www.w3.org/1999/XMLSchema-instance'
  xmlns:xsd='http://www.w3.org/1999/XMLSchema'
  xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'>

  <soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
    <soap:Body>
      <n:getRateResponse xmlns:n='urn:xmethods-CurrencyExchange'>
        <Result xsi:type='xsd:float'>1.6059</Result>
      </n:getRateResponse>
    </soap:Body>
  </soap:Envelope>

```

The result of this Remote Procedure Call (1.6059 UK pounds to one US dollar in this example) is returned as the result of the Web Services object.

Web Services Resources

Consult these resources to get more information about Web Services in general and related technologies:

- <http://msdn.microsoft.com/webservices/>
- http://en.wikipedia.org/wiki/Web_service
- <http://www.w3.org>



Appendix

C

URS-Behind Solution

This Appendix includes the following sections:

- [URS-Behind Solution Introduction, page 797](#)
- [Message Formats, page 798](#)
- [Supported Interfaces, page 800](#)
- [Supported Methods, page 811](#)

URS-Behind Solution Introduction

The major goal of the URS-Behind solution is to provide the resource allocation services, directly or through the Web, which are implemented in Universal Routing Server (URS) to interested parties.

It is the responsibility of the Client (Web client) to handle the services with the selected resource, for example, deciding to transfer calls to the agent that URS has provided.

Additionally, URS-Behind Solution is extended to provide different types of statistics, metrics, performance, troubleshooting, and other information that URS has in its possession.

To utilize this functionality, the client either connects to URS directly (for example through Genesys PSDK), or through an HTTP Interface component which supports HTTP protocols such as REST or SOAP. This frees the client from the necessity of knowing in-depth, low level details about the protocols used by the URS-Behind Solution, as described in the "Message Formats" on [page 798](#).

- With a direct connection to URS, the client uses a low level TCP/IP connection and implements it as described in "Message Formats" protocol. Or, alternatively, the client utilizes Genesys PSDK tools which already implement this protocol. Direct connection can be configured on any URS ports except those used by HTTP Bridge and/or HTTP Interface components.

- See Web Services/Web API Options, [page 687](#).
- Connecting through an HTTP Interface is accessible from any Web client. It requires enabling of Web API functionality.
See “Web API (URS-Behind) Options” on [page 692](#)

Message Formats

Messages to request and receive the routing service use the standard Genesys KVList format. They define the "language" used by the clients directly connecting to URS. Specifically, both Genesys PSDK and HTTP Interface modules use them internally.

R-COMMAND MESSAGES

The generic format of R-Command messages is a KVList with the following structure:

```

Key R_MSG value KVList:
Key R_TYPE value string: Request, Trying, OK, Error
Key R_REF value integer: reference id to map requests and responses
Key R_BODY value KVList:
Key Method value KVList:
Key Name value string : name of method
Key Params value KVList:
    Key <param_name> value string
    Key <param_name> value string
.

    Key CustomerID value string: tenant name
Key Context value KVList:
    Key <context_key> value <context_key_value>
    Key <context_key> value <context_key_value>
.

    Key Result value KVList:
        Key <result_key> value <result_key_value>
        Key <result_key> value <result_key_value>
.
.
```

Notes:

- R_TYPE, R_REF, and R_BODY are *mandatory* for all command messages except for Trying, which contains only R_TYPE and R_REF.
- A command message from a client to the URS must be a Request. URS will answer with either the OK or Error command message.
- The value of R_BODY for a Request message must contain Method, and can optionally contain Context and CustomerID.

- The value of `R_MESSAGE_BODY` for an `OK` message must contain `Result`, and can optionally contain `Context`.
- The specific content of `Parameters` and `Result` depends on the `Method` used.
- `Context` is used for methods that operate on interactions. It provides information about interactions, can be changed by the executed method, and, therefore, can be present in both `Request` and `OK`. For example, if the method is `RunStrategy`, then it can update user data attached to a call while it is processed by a strategy. Currently, interaction context imitates `TEvents` and includes the following context keys:
 - Key `CallType`, value type of call, integer.
 - Key `ThisQueue`, value this queue of call, string.
 - Key `OtherDN`, value Other DN of call, string.
 - Key `ANI`, value ANI of call, string.
 - Key `DNIS`, value DNID of call, string.
 - Key `CED`, value CED queue of call, string.
 - Key `MediaType`, value Media Type of call, string.
 - Key `UserData`, value of call attached data (`KVList`).
 - Key `Extensions`, value of call extensions (`KVList`).
- The `Trying` message indicates that URS started processing the request. It will be followed later with either the `OK` or `Error` message. It is possible that the `Error` message will be answered without preliminary `Trying`.
- Beginning with URS 8.1.4, for web requests, URS does not take the option `ignore_customer_id` into account and works as if this option is set to `false`.

R-COMMAND-CONTROL MESSAGES

In addition to command messages, a set of messages is designed to control the execution of command messages. They have a different, though similar, format and their purpose is to query the execution status of command messages, to terminate execution of command messages, and so on.

The generic format of R-Command-Control messages is a `KVList` with the following structure:

```

Key R_CTL_MSG, value KVList:
Key R_TYPE, value string: Request, OK, Error
Key R_REF, value integer: reference id to map requests and responses
Key R_CTL_REF, value integer: reference id of the queried command
message
Key R_CTL_CMD, value string: controlling action · Abort, Query
  
```

Notes:

- All attributes are mandatory for Request control messages. `R_CTL_CMD` is optional for `OK` or `Error`.

- Abort control message results in cancellation of the request referenced by R_REF. OK is returned when the request was found and aborted. Error indicates that the request was not found (does not exist).
- Query control message results in lookup of the request referenced by R_REF. OK is returned when the request was found. Error indicates that the request was not found (does not exist).

Supported Interfaces

URS supports a number of interfaces that can be used to invoke URS-Behind functionality. They are described below.

Direct Connection to URS

A client can use a URS-Behind solution by connecting to URS directly using the URS library API. In this case, the client is built *on top of* the Genesys connection library.

The URS library now supports passing and processing of messages described in the earlier section “Message Formats” on [page 798](#). The following sections describe related data types, and provide basic use cases.

Data Types

The following code fragment includes data types from the URS library used to support URS-Behind functionality.

```
typedef enum
{
    ...
    rRejected,
    rDefaultRouting,
    rNoCall
} rErrorCode;

typedef enum
{
    rGeneric = 0,
    ...
} rClientType;

typedef enum
{
    ...
    R_command_message,
    R_result,
```

```

    ...
} rAttrName;

typedef enum
{
...
rRunCommand,
...
rCommand = 100,
rCtlCommand,
...
} rClientRequest;

typedef enum
{
...
rCommandTrying = 199,
rCommandResponse = 200,
...
} rEventType;

```

Use Cases

Connecting to URS

URS performs client registration when a client connects to it. The registration procedure is as follows:

```

Client -> rRegisterClient(R(userID = "<clientName>",
R_client_type=rGeneric)
-> Router
Client <- rOpenOK(R_client_number=socket, R_ErrorCode=rNoError) <-
Router

```

Note: URS will work with a client even if it does not register.

URS Shutdown

Every client gets a notification on URS shutdown (when URS exits normally):

```
Client <- rShutdown() <-Router
```

Requesting Service

A client can send any R-Command or R-Command-Control message to URS. Possible scenarios are described below.

- Success:

- ```

Client -> rRunCommand(R_refID=<n>, R_command_message=R-Command
Request) -> Router
Client <- rInfoMessage(R_refID=<n>, R_ErrorCode=rNoError,
R_result=R-Command Trying) <- Router
Client <- rInfoMessage(R_refID=<n>, R_ErrorCode=rNoError,
R_result=R-Command OK) <- Router

```
- Failure to understand the request (if this is an older URS version or an unknown R-Command message):
 

```

Client -> rRunCommand(R_refID=<n>, R_command_message=R-Command
Request) -> Router
Client <- rErrorMessage(R_refID=<n>, R_ErrorCode=rNotAvailable) <- Router

```
  - Failure to start execution (for example, if no strategy to run was found):
 

```

Client -> rRunCommand(R_refID=<n>, R_command_message=R-Command
Request) -> Router
Client <- rInfoMessage(R_refID=<n>, R_ErrorCode=rNoError,
R_result=R-Command Error) <- Router

```
  - Processing failure (for example, on default routing):
 

```

Client -> rRunCommand(R_refID=<n>, R_command_message=R-Command
Request) -> Router
Client <- rInfoMessage(R_refID=<n>, R_ErrorCode=rNoError,
R_result=R-Command Trying) <- Router
Client <- rInfoMessage(R_refID=<n>, R_ErrorCode=rNoError,
R_result=R-Command Error) <- Router

```

### Requesting Service (Short Form)

Instead of passing R-Command messages in their entirety (as specified in “R-COMMAND MESSAGES” on [page 798](#)), a client can pass only R\_BODY content and place all top-level elements directly in message attributes.

- Success:
 

```

Client -> rCommand(R_refID=<n>, R_command_message=R_BODY Content
Request) -> Router
Client <- rCmdTrying(R_refID=<n>, R_ErrorCode=rNoError) <- Router
Client <- rCmdResponse(R_refID=<n>, R_ErrorCode=rNoError, R_result=
OK R_BODY) <- Router

```
- Failure to understand the request (if this is an older URS version or an unknown R-Command message):
 

```

Client -> rRunCommand(R_refID=<n>, R_command_message=R-Command
Request) -> Router
Client <- rErrorMessage(R_refID=<n>, R_ErrorCode=rNotAvailable) <- Router

```
- Failure to start execution (for example, if no strategy to run was found):
 

```

Client -> rRunCommand(R_refID=<n>, R_command_message=R-Command
Request) -> Router

```

```

Client <- rErrorMessage (R_refID=<n>, R_ErrorCode=rRejected,
R_result=error R_BODY) <-Router
• Processing failure (for example, on default routing):
Client -> rRunCommand(R_refID=<n>, R_command_message=R-Command
Request) -> Router
Client <- rCmdTrying(R_refID=<n>, R_ErrorCode=rNoError) <- Router
Client <- rErrorMessage(R_refID=<n>, R_ErrorCode=rDefaultRouting
R_result=Error R_BODY) <- Router
• Success processing a control message (the request referenced by the
control message was found or aborted):
Client -> rCtlCmd(R_refID=<n>, R_command_message=[R_CTL_CMD
R_CTL_REF] KVList) -> Router
Client <- rCmdResponse(R_refID=<n>, R_ErrorCode=rNoError) <- Router
• Failure processing a control message (the request referenced by the control
message was not found):
Client -> rCtlCmd(R_refID=<n>, R_command_message=[R_CTL_CMD
R_CTL_REF] KVList) -> Router
Client <- rErrorMessage(R_refID=<n>, R_ErrorCode=rNoCall) <- Router

```

## SOAP/Plain HTTP Interface

The HTTP Interface component within URS enables URS-Behind functionality as a web service and supports both SOAP and REST protocols.

The content of the response `<Result>` element for target-selection requests (`RunStrategy`, `getTarget`) is the same as in extensions of a `RequestRouteCall` event when a similar request is received as a regular `EventRouteRequest` event from T-Server, with these exceptions:

- If a service request results in default routing, then `<Result>` will additionally contain key `Reason` with value `Default`.
- If a service request results in rejection of routing, then `<Result>` will additionally contain key `Reason` with value `Rejected`.

There is no `Reason` key in `<Result>` when a regular target is returned (if the strategy does not explicitly insert this key).

Further details for both SOAP and plain HTTP interface are described in the following sections.

## SOAP Interface

Formally, SOAP Interface methods are specified in the referred GUR.wsdl file. The `GUR.wsdl` file describes the SOAP interface to URS. The `GUR.wsdl` file is included in the URS installation package, and the content is also provided below:

```

<?xml version="1.0" encoding="UTF-8"?>
<definitions name="GUR"

```

```

targetNamespace="http://www.genesyslab.com/ur"
xmlns:tns="http://www.genesyslab.com/ur"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:gsd="http://www.genesyslab.com/cs"
xmlns:gur="http://www.genesyslab.com/ur"
xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:MIME="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:DIME="http://schemas.xmlsoap.org/ws/2002/04/dime/wsdl/"
xmlns:WSDL="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/">>

<types>

<schema targetNamespace="http://www.genesyslab.com/ur"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:gsd="http://www.genesyslab.com/cs"
xmlns:gur="http://www.genesyslab.com/ur"
xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
<import namespace="http://www.genesyslab.com/cs"
schemaLocation="http://www.genesyslab.com/cs"/>
<import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
<complexType name="ContextKeys">
<annotation>
<documentation>Predefined context keys</documentation>
</annotation>
<sequence>
<element name="CallType" type="xsd:string" minOccurs="0"
maxOccurs="1"
nillable="true" default="CallType"/>
<element name="ThisQueue" type="xsd:string" minOccurs="0"
maxOccurs="1"
nillable="true" default="ThisQueue"/>
<element name="OtherDN" type="xsd:string" minOccurs="0"
maxOccurs="1"
nillable="true" default="OtherDN"/>
<element name="ANI" type="xsd:string" minOccurs="0"
maxOccurs="1"
nillable="true" default="ANI"/>
<element name="DNIS" type="xsd:string" minOccurs="0"
maxOccurs="1"
nillable="true" default="DNIS"/>
<element name="CED" type="xsd:string" minOccurs="0"
maxOccurs="1"
nillable="true" default="CED"/>

```

```

 nillable="true" default="CED"/>
 <element name="MediaType" type="xsd:string" minOccurs="0"
maxOccurs="1"
 nillable="true" default="MediaType"/>
 <element name="UserData" type="xsd:string" minOccurs="0"
maxOccurs="1"
 nillable="true" default="UserData"/>
 <element name="Extensions" type="xsd:string" minOccurs="0"
maxOccurs="1" nillable="true" default="Extensions"/>
 </sequence>
 </complexType>
 <!-- operation request element -->
 <element name="RunStrategy">
 <complexType>
 <sequence>
 <element name="Strategy" type="xsd:string" minOccurs="1"
maxOccurs="1"
 nillable="false"/>
 <element name="CustomerID" type="xsd:string" minOccurs="1"
maxOccurs="1" nillable="false"/>
 <element name="Context" type="gsd:list_pair" minOccurs="0"
maxOccurs="1" nillable="true"/>
 </sequence>
 </complexType>
 </element>
 <!-- operation response element -->
 <element name="StrategyResponse">
 <complexType>
 <sequence>
 <element name="Result" type="gsd:list_pair" minOccurs="1"
maxOccurs="1"
 nillable="false"/>
 <element name="Context" type="gsd:list_pair" minOccurs="0"
maxOccurs="1" nillable="true"/>
 </sequence>
 </complexType>
 </element>
 <!-- operation request element -->
 <element name="RunStrategy2">
 <complexType>
 <sequence>
 <element name="Strategy" type="xsd:string" minOccurs="1"
maxOccurs="1"
 nillable="false"/>
 <element name="CustomerID" type="xsd:string" minOccurs="1"
maxOccurs="1" nillable="false"/>
 <element name="CallType" type="xsd:int" minOccurs="0"
maxOccurs="1"/>
 <element name="ThisQueue" type="xsd:string" minOccurs="0"
maxOccurs="1"
 nillable="true"/>
 </sequence>
 </complexType>
 </element>

```

```

 <element name="OtherDN" type="xsd:string" minOccurs="0"
maxOccurs="1"
 nillable="true"/>
 <element name="ANI" type="xsd:string" minOccurs="0"
maxOccurs="1"
 nillable="true"/>
 <element name="DNIS" type="xsd:string" minOccurs="0"
maxOccurs="1"
 nillable="true"/>
 <element name="CED" type="xsd:string" minOccurs="0"
maxOccurs="1"
 nillable="true"/>
 <element name="MediaType" type="xsd:string" minOccurs="0"
maxOccurs="1"
 nillable="true"/>
 <element name="UserData" type="gsd:list_pair" minOccurs="0"
maxOccurs="1" nillable="true"/>
 <element name="Extensions" type="gsd:list_pair" minOccurs="0"
maxOccurs="1" nillable="true"/>
 </sequence>
</complexType>
</element>
<!-- operation request element -->
<element name="GetTarget">
 <complexType>
 <sequence>
 <element name="CustomerID" type="xsd:string" minOccurs="1"
maxOccurs="1" nillable="false"/>
 <element name="TargetList" type="xsd:string" minOccurs="1"
maxOccurs="1" nillable="false"/>
 <element name="WaitTime" type="xsd:int" minOccurs="0"
maxOccurs="1"/>
 <element name="Statistic" type="xsd:string" minOccurs="0"
maxOccurs="1"
 nillable="true"/>
 <element name="UseMinStatValue" type="xsd:int" minOccurs="0"
maxOccurs="1"/>
 <element name="VirtualQueue" type="xsd:string" minOccurs="0"
maxOccurs="1" nillable="true"/>
 <element name="Priority" type="xsd:int" minOccurs="0"
maxOccurs="1"/>
 <element name="MediaType" type="xsd:string" minOccurs="0"
maxOccurs="1"
 nillable="true"/>
 </sequence>
 </complexType>
</element>
<!-- operation request element -->
<element name="GetStatistics">
 <complexType>
 <sequence>

```

```

<element name="CustomerID" type="xsd:string" minOccurs="1"
 maxOccurs="1" nillable="false"/>
<element name="TargetList" type="xsd:string" minOccurs="1"
 maxOccurs="1" nillable="false"/>
<element name="Statistic" type="xsd:string" minOccurs="1"
maxOccurs="1"
 nillable="false"/>
</sequence>
</complexType>
</element>
</schema>

</types>

<message name="RunStrategy">
 <part name="parameters" element="gur:RunStrategy"/>
</message>

<message name="StrategyResponse">
 <part name="parameters" element="gur:StrategyResponse"/>
</message>

<message name="RunStrategy2">
 <part name="parameters" element="gur:RunStrategy2"/>
</message>

<message name="GetTarget">
 <part name="parameters" element="gur:GetTarget"/>
</message>

<message name="GetStatistics">
 <part name="parameters" element="gur:GetStatistics"/>
</message>

<portType name="GURSoapPort">
 <operation name="RunStrategy">
 <documentation>RunStrategy method</documentation>
 <input message="tns:RunStrategy"/>
 <output message="tns:StrategyResponse"/>
 </operation>
 <operation name="RunStrategy2">
 <documentation>RunStrategy2 method</documentation>
 <input message="tns:RunStrategy2"/>
 <output message="tns:StrategyResponse"/>
 </operation>
 <operation name="GetTarget">
 <documentation>GetTarget method</documentation>
 <input message="tns:GetTarget"/>
 <output message="tns:StrategyResponse"/>
 </operation>
 <operation name="GetStatistics">

```

```

<documentation>GetStatistics method</documentation>
<input message="tns:GetStatistics"/>
<output message="tns:StrategyResponse"/>
</operation>
</portType>

<binding name="GURSoapBinding" type="tns:GURSoapPort">
<SOAP:binding style="document"
 transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="RunStrategy">
 <SOAP:operation
 soapAction="http://genesyslab.com/ur/action/UR.RunStrategy"/>
 <input>
 <SOAP:body parts="parameters" use="literal"/>
 </input>
 <output>
 <SOAP:body parts="parameters" use="literal"/>
 </output>
</operation>
<operation name="RunStrategy2">
 <SOAP:operation
 soapAction="http://genesyslab.com/ur/action/UR.RunStrategy2"/>
 <input>
 <SOAP:body parts="parameters" use="literal"/>
 </input>
 <output>
 <SOAP:body parts="parameters" use="literal"/>
 </output>
</operation>
<operation name="GetTarget">
 <SOAP:operation
 soapAction="http://genesyslab.com/ur/action/UR.GetTarget"/>
 <input>
 <SOAP:body parts="parameters" use="literal"/>
 </input>
 <output>
 <SOAP:body parts="parameters" use="literal"/>
 </output>
</operation>
<operation name="GetStatistics">
 <SOAP:operation
 soapAction="http://genesyslab.com/ur/action/UR.GetStatistics"/>
 <input>
 <SOAP:body parts="parameters" use="literal"/>
 </input>
 <output>
 <SOAP:body parts="parameters" use="literal"/>
 </output>
</operation>
</binding>
```

```

<service name="GUR">
 <documentation>Genesys Universal Routing</documentation>
 <port name="GURSoapPort" binding="tns:GURSoapBinding">
 <SOAP:address location="REAL_ADDRESS_TO_BE_CHANGED"/>
 </port>
</service>

</definitions>

```

Here is an example of a SOAP request to Router:

```

<?xml version="1.0"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:gsd="http://www.genesyslab.com/cs"
 xmlns:gur="http://www.genesyslab.com/ur">

 <SOAP-ENV:Body>
 <gur:RunStrategy>
 <gur:Strategy>Single_sw1_Q8111-8119_108b</gur:Strategy>
 <gur:CustomerID>Gadgets</gur:CustomerID>
 <gur:Context xsi:type="gsd:list_pair">
 <int_pair xsi:type="gsd:int_pair" key="CALLTYPE"
 value="1"/>
 <str_pair xsi:type="gsd:str_pair" key="ThisQueue" value="2201">
 </str_pair>
 <str_pair xsi:type="gsd:str_pair" key="OtherDN"
 value="2202"/>
 <str_pair xsi:type="gsd:str_pair" key="ANI" value="2015675678">
 </str_pair>
 <str_pair xsi:type="gsd:str_pair" key="DNIS"
 value="333"/>
 <str_pair xsi:type="gsd:str_pair" key="CED"
 value="5"/>
 <str_pair xsi:type="gsd:str_pair" key="MediaType" value="voice">
 </str_pair>
 <list_pair xsi:type="gsd:list_pair" key="UserData">
 <str_pair xsi:type="gsd:str_pair" key="data"
 value="attachment_0"/>
 </str_pair>
 <int_pair xsi:type="gsd:int_pair" key="user_data1"
 value="182"/>
 </int_pair>
 <str_pair xsi:type="gsd:str_pair" key="user_data2"
 value="test_2"/>
 </str_pair>
 </list_pair>
 </gur:Context>
 </gur:RunStrategy>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

```
</gur:RunStrategy>
</SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

Here is an example of a SOAP response when processing is completed normally (HTTP 200 OK):

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:gsd="http://www.genesyslab.com/cs"
 xmlns:gur="http://www.genesyslab.com/ur">

 <SOAP-ENV:Body>
 <gur:StrategyResponse>
 <gur:Result xsi:type="gsd:list_pair">
 <str_pair xsi:type="gsd:str_pair" key="Reason" value="Rejected">
 </str_pair>
 </gur:Result>
 <gur:Context xsi:type="gsd:list_pair">
 <list_pair xsi:type="gsd:list_pair" key="UserData">
 <str_pair xsi:type="gsd:str_pair" key="data"
 value="attachment_0">
 </str_pair>
 <int_pair xsi:type="gsd:int_pair" key="user_data1"
 value="182">
 </int_pair>
 <str_pair xsi:type="gsd:str_pair" key="user_data2"
 value="test_2">
 </str_pair>
 <int_pair xsi:type="gsd:int_pair" key="PegRejected" value="1">
 </int_pair>
 </list_pair>
 </gur:Context>
 </gur:StrategyResponse>
 </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Here is an example of a SOAP response when processing has failed (HTTP 500 Internal Server Error):

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:gsd="http://www.genesyslab.com/cs"
```

```
xmlns:gur="http://www.genesyslab.com/ur">

<SOAP-ENV:Body
 SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Fault>
 <faultcode>SOAP-ENV:Client</faultcode>
 <faultstring>Method 'echo' not implemented: method name or
namespace
 not recognized
 </faultstring>
 </SOAP-ENV:Fault>
</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

## Plain HTTP Interface

The HTTP Interface implements the REST-based URS Web Service. Any command can be sent with either GET or POST HTTP request. In a POST HTTP request, the body of the POST message should have application/x-www-form-urlencoded Content-Type.

The REST Web methods can be classified into five different groups to run strategies:

- run strategies
  - get available targets
  - get statistics
  - diagnostic/performance metrics
  - help/method descriptions

See the *URS 8.1 Deployment Guide*: Chapter 1: Routing Overview, URS Web Interface, Figure: URS Web Interface.

The REST Interface does not have a formal description of its method. Instead, samples of some of the methods to run strategies are provided in the following Supported Methods section.

# **Supported Methods**

This section provides samples of methods to run strategies for the SOAP method and the Plain HTTP method.

# **SOAP Methods**

As described below, the SOAP Methods include the following:

- RunStrategy
- GetTarget
- GetStatistics

## RunStrategy

Name: RunStrategy

Parameters: Strategy, string value – name of the strategy to execute  
CustomerID – name of the tenant  
Context – KVList value (see “R-Command Messages” section above)

This method runs the specified strategy and is a generic analogy of the EventRouteRequest event. It requires a CustomerID value in R\_BODY since the strategy will be running within the specified tenant. When this method is invoked, one of the following results can be expected:

- OK message is returned when the result is successful. Target for the interaction was found (other than the default target). Result contains extensions of the interaction (equal to the content of AttributeExtensions of the RequestRouteCall event.) URS automatically puts information about the target the call was routed to into extensions (keys AGENT, PLACE, VQ, and so on). Also, the strategy can put other data into extensions.
- Error message is returned when the result is negative (this includes selection of the default target), whether because the strategy could not be started, or routing was rejected explicitly while the strategy was running (function Exit was called), or the target was not found (function Default was called). Result can contain a Reason key explaining the error (current values: Rejected, Default).

## GetTarget

Name: GetTarget

Parameters: CustomerID – name of the tenant  
TargetList, string value – comma separated list of targets  
WaitTime, integer value – time to wait in seconds  
Statistic, string value – name of the statistic to select the target  
UseMinStatValue, integer value – 1: select max, 2: select min.  
VirtualQueue, string value – name of the virtual queue  
Priority, integer value – priority of the request  
MediaType – media type

This method imitates execution of one Target Selection object in a strategy. It behaves identically with RunStrategy method but uses a hard-coded strategy.

When this method is invoked, the results are the same as those for the RunStrategy method.

## GetStatistics

Name: **GetStatistics**

Parameters: **CustomerID** – name of the tenant

**TargetList**, **string value** – comma separated list of targets

**Statistic**, **string value** – name of the requested statistic

This method requests URS to return values of the requested statistic for all targets. When this method is invoked, one of the following results can be expected:

- OK message is returned when the result is successful. Result contains a key-value pair for each target with the target as the pair key and the value of the statistic as the pair value.

Error message is returned when the result is negative.

## Plain HTTP Methods

The Plain HTTP Methods include the following, as described in this section:

- RunStrategy2
- start
- GetTargets
- GetStatistics
- sdata
- rvqdata
- query
- report
- console?groups
- help

## RunStrategy2

This method is a duplication of the SOAP RunStrategy method. It runs the strategy and returns the data.

```
http://host:port/RunStrategy2?strategy=xyz&CustomerID=abc&CallType=1&OtherDN=5678&MediaTypes=voice&ANI=12345&DNIS=180012345678&CED=12345&UserData=aaaa:1|bbbb:2
```

---

**Notes:**

- The value of the CallType parameter is presented as a number according to the corresponding TLibrary enumerator (from 0 to 4). The value of MediaType is a string representing the media type, as defined in the Configuration Server media type enumerator (voice, and so on).
  - This method returns data in SOAP format just like its SOAP counterpart RunStrategy method.
- 

**start**

This method has no SOAP counterpart. It starts a new strategy instance and returns its ID.

```
http://host:port/urs/call/start?strategy=strategynname&ani=12345&dnis=18
0012345678&tenant=tenantname&udata.key1=value1&..¶m1=value1&
param2=value2
```

**GetTarget**

This method is a duplication of the SOAP GetTarget method. It imitates execution of one Target Selection object in a strategy that is provided in the request parameters.

```
http://host:port/GetTarget?CustomerID=abc&TargetList=agent1.A, agent2.A&
WaitTime=60&Statistic=StatTimeInReadyState&UseMinStatValue=0&VirtualQue
ue=r1234&Priority=0
```

---

**Note:** This method returns data in the SOAP format, just like its SOAP counterpart GetTarget method.

---

**exec**

This method is effectively the same as the GetTarget method, however it returns data in JSON format, not in SOAP.

Sample:

```
http://host:port/urs/call/exec?tenant=abc&TargetList=
agent1.A, agent2.A&WaitTime=600&Statistic=StatTimeInReadyState&Criteria=
1&VirtualQueue=r1234&Priority=10
```

**GetStatistics**

This method is a duplication of the SOAP GetStatistics method. It returns the value of the requested statistics for all listed targets.

Sample:

```
http://host:port/GetStatistics?CustomerID=abc&TargetList=agent1.A, agent2.A&Statistic=StatTimeInReadyState
```

---

**Note:** This method returns data in SOAP format just like its SOAP counterpart GetStatistics method.

---

## sdata

This method is effectively the same as the SOAP GetStatistics method however, it returns data in JSON format, not in SOAP.

Sample:

```
http://host:port/urs/stat/sdata?tenant=abc&targets=agent1.A, agent2.A&statistic=StatTimeInReadyState
```

## rvqdata

This method provides REST-based access to the RvqData function ([page 570](#)). It accepts only the label of the internal routing queue, and returns all metrics (except RVQ\_DATA\_MIN\_EWT and RVQ\_DATA\_AGR\_EWT) in the response.

Sample:

```
http://host:port/urs/call/<interaction identification>/rvqdata?id=labe
```

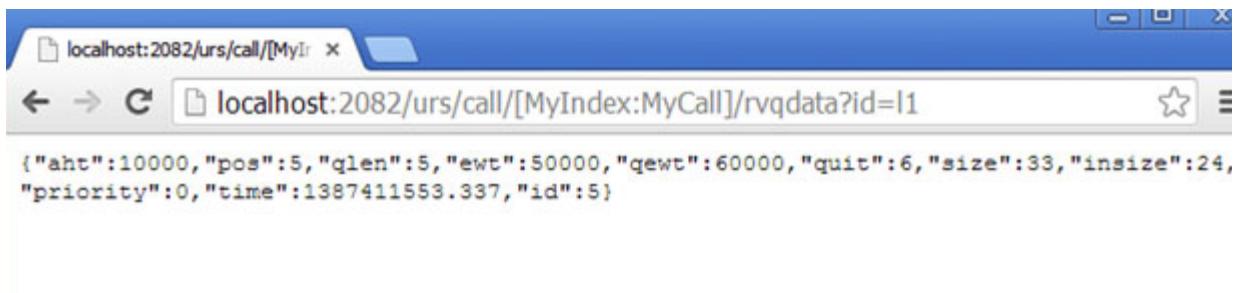


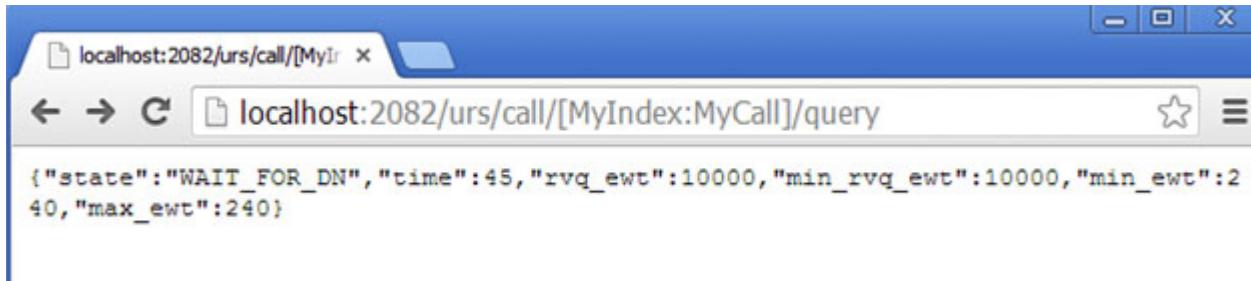
Figure 272: Description of rvqdata Method

## query

This method provides REST-based access to query an existing interaction. It returns the interaction state and RvqData function metrics which are not provided with the rvqdata method.

Sample:

```
http://host:port/urs/call/<interaction identification>/query
```



**Figure 273: Description of Query Method**

## report

This method provides generic information, such as:

- how many interactions were processed by the Router
- how many interaction were processed by a specific T-Server or Routing Point
- agent reservation requests usage
- URS licenses usage

To provide information about processed interactions

**urs/stat/report?tserver=name&cdn=digits&count**

Input:

- tserver: optional - name of T-Server for which information is required.
- cdn: optional - number of routing point for which information is required.
- count: optional - if present, URS reports only high-level information (no time based counters).

Sample:

<http://localhost:2082/urs/stat/report?tserver= ts2&cdn=2203>

will result in:

```
<statinfo>
 <tserver name="ts2" switch="sw2">
 <cdn digits="2203">
 <now>0</now>
 <handled>
 <calls>4</calls>
 <time total="156" n="23" t="0" x="0" s="0" w="117"
 f="0" r="11" />
 </handled>
 </tserver>
</statinfo>
```

```

<routed>
 <calls>4</calls>
 <time total="156" n="23" t="0" x="0" s="0" w="117"
 f="0" r="11" />
</routed>
<confirmed>
 <calls>4</calls>
 <time total="156" n="23" t="0" x="0" s="0" w="117"
 f="0" r="11" />
</confirmed>
<abandoned><calls>0</calls></abandoned>
<error><calls>0</calls></error>
<default><calls>0</calls></default>
<dropped><calls>0</calls></dropped>
</cdn>
</tserver>
</statinfo>

```

To provide information about agent reservation requests usage

**urs/stat/report?ar**

Sample:

<http://localhost:2082/urs/stat/report?ar>

will result in:

```

<agent_reservation_info>
 <exp_reqs>2</exp_reqs>
 <imp_reqs>0</imp_reqs>
 <pos>2</pos>
 <neg>0</neg>
 <used>2</used>
</agent_reservation_info>

```

To provide information about router\_seats, specifically, how many of them were used

**urs/stat/report?seats**

Sample:

<http://host:port/urs/stat/report?seats>

will result in:

```
<router_seats>
```

```

<used>2</used>
<max>0</max>
</router_seats>
```

## console?groups

This method provides the possibility to explore content statistics (group, skill expressions) that URS has in its memory.

- return the number of content statistics

Sample:

```
http://host:port/urs/console?groups * #
```

- return the list of all opened group statistics

Sample:

```
http://host:port/urs/console?groups * *
```

- return the number of content statistics that use the skill French.

Sample:

```
http://host:port/urs/console?groups French #
```

- return the list of all content statistics that use the skill French.

Sample:

```
http://host:port/urs/console?groups French *
```

```

<?: B>7-__1 & C>3 & C<8 & { B>0 | C>0 }> size 0
<?: russian>3-__1 & russian<8+__1 & { russian>0 }> size 0
<?: B>7-__1 & C>3 & C<8 & { B>0 | C>0 }> size 0
<?: russian>3-__1 & russian<8+__1 & { russian>0 }> size 0
<?Agents1_5: French>3-__1 & French<8+__1 & { French>0 }> size 5
<?Agents1_5:> size 5
<?Agents_sw2: French>7-__1 & { French>0 }> size 10
<?Agents_sw2:> size 29

```

## Description of Report Method - Console Groups

### help

This method provides a short description of some of the REST API methods like query, rvqdata, and report.

Samples:

```
http://host:port/urs/help/call/query
```

`http://host:port/urs/help/call/rvqdata`  
`http://host:port/urs/help/stat/report.`

---

**Note:** The http port and the soap port can also be configured in the Server Info tab of the router Application object.

---





## Supplements

# Related Documentation Resources

The following resources provide additional information that is relevant to this software. Consult these additional resources as necessary.

## Universal Routing

- *Universal Routing 8.1 Deployment Guide*, which describes installation, configuration, and deployment of the Universal Routing Server and Custom Server components.
- *Universal Routing 8.1 Business Process User's Guide*. This guide contains step-by-step instructions for creating interaction workflows (business processes), which direct incoming customer interactions through various processing objects. The goal is to generate an appropriate response for the customer.
- *Universal Routing 8.1 Strategy Samples*, which simplifies strategy configuration for first-time users of the strategy development tool, Interaction Routing Designer. To achieve this goal, this document supplies examples of simple voice and e-mail routing strategies that can be used as general guides during the design stage.
- *Universal Routing 8.0 Routing Application Configuration Guide* (previously *Universal Routing 7.0 Routing Solutions Guide*), which contains information on the various types of routing that can be implemented, including skills-based routing, business-priority routing, and share agent by service level agreement routing. It also summarizes cost-based routing, proactive routing, and a SIP/instant message solution.
- *Universal Routing 7.6 (or later) Cost-Based Routing Configuration Guide*, which documents how to configure Universal Routing Server to use the cost of routing to a target, consisting of Infrastructure cost and/or Resource cost, as addition selection criteria when choosing the right target.

- *Universal Routing 8.1 Interaction Routing Designer Help*, which describes how to use Interaction Routing Designer to create routing strategies. It also describes Interaction Design view where you create business processes that route incoming interactions through various processing objects with the goal of generating an appropriate response for the customer.

## eServices and Other

- *eServices (Multimedia) 8.1 Deployment Guide*, which includes a high-level overview of features and functions of Genesys eServices together with architecture information and deployment-planning materials. It also introduces you to some of the basic concepts and terminology used in this product.
- *eServices 8.1 User's Guide*, which provides overall information and recommendations on the use and operation of Genesys eServices.
- *eServices 8.1 Open Media Interaction Models Reference Manual*, which presents a set of basic interaction models, showing the components involved and the messaging (requests, events) among them.
- “Universal Routing and Multimedia/eServices Log Events” in *Framework Combined Log Events Help*, which is a comprehensive list and description of all events that may be recorded in Management Layer logs.
- *Orchestration Server 8.1 Deployment Guide*, which describes installation, configuration, and deployment of the Genesys Orchestration Server, which works with the Universal Routing Server.

## Genesys

- *Genesys 8.1 Proactive Routing Solution Guide*, which documents a solution that enables you to proactively route outbound\_preview interactions to Genesys Agent Desktop, as well as to completely process Calling List and Do Not Call List records solely from the logic of a routing strategy without agent intervention.
- *Genesys Events and Models Reference Manual*, which provides information on most of the published Genesys events and their attributes, and an extensive collection of models describing core interaction processing in Genesys environments.
- *Genesys Technical Publications Glossary*, which provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms used in this document.
- *Genesys Migration Guide*, which ships on the Genesys Documentation Library DVD, and which provides documented migration strategies for Genesys product releases. Contact [Genesys Customer Care](#) for more information.

- Release Notes and Product Advisories for this product, which are available on [docs.genesys.com/Documentation/Routing](http://docs.genesys.com/Documentation/Routing), and on the Genesys Customer Care website at <http://genesys.com/customer-care>.

Information about supported hardware and third-party software is available on the Genesys Documentation website in the following documents:

- [\*Genesys Supported Operating Environment Reference Guide\*](#)
- [\*Genesys Supported Media Interfaces Reference Manual\*](#)

Additional System Level Guide resources are available on [docs.genesys.com](http://docs.genesys.com):

- *Genesys Hardware Sizing Guide*, which provides information about Genesys hardware sizing guidelines for the Genesys 7.x and Genesys 8.x releases.
- *Genesys Interoperability Guide*, which provides information on the compatibility of Genesys products with various Configuration Layer Environments; Interoperability of Reporting Templates and Solutions; and Gplus Adapters Interoperability.
- *Genesys Licensing Guide*, which introduces you to the concepts, terminology, and procedures relevant to the Genesys licensing system.
- *Genesys Database Sizing Worksheets*, which provides a range of expected database sizes for various Genesys products.

For additional system-wide planning tools and information, see the release-specific listings of System Level Documents on the [Genesys Documentation](#) website.

Genesys product documentation is available on the:

- Genesys Documentation website at <http://docs.genesys.com>
- Genesys Documentation Library DVD, which you can order by e-mail from Genesys Order Management at [orderman@genesys.com](mailto:orderman@genesys.com).

# Document Conventions

This document uses certain stylistic and typographical conventions—introduced here—that serve as shorthands for particular kinds of information.

## Document Version Number

A version number appears at the bottom of the inside front cover of this document. Version numbers change as new information is added to this document. Here is a sample version number:

81r\_ref\_04-2011\_v8.1.001.00

You will need this number when you are talking with Genesys Customer Care about this product.

## Screen Captures Used in This Document

Screen captures from the product graphical user interface (GUI), as used in this document, may sometimes contain minor spelling, capitalization, or grammatical errors. The text accompanying and explaining the screen captures corrects such errors *except* when such a correction would prevent you from installing, configuring, or successfully using the product. For example, if the name of an option contains a usage error, the name would be presented exactly as it appears in the product GUI; the error would not be corrected in any accompanying text.

## Type Styles

[Table 144](#) describes and illustrates the type conventions that are used in this document.

**Table 144: Type Styles**

Type Style	Used For	Examples
Italic	<ul style="list-style-type: none"> <li>Document titles</li> <li>Emphasis</li> <li>Definitions of (or first references to) unfamiliar terms</li> <li>Mathematical variables</li> </ul> <p>Also used to indicate placeholder text within code samples or commands, in the special case where angle brackets are a required part of the syntax (see the note about angle brackets on page 825).</p>	<p>Please consult the <i>Genesys Migration Guide</i> for more information.</p> <p>Do <i>not</i> use this value for this option.</p> <p>A <i>customary and usual</i> practice is one that is widely accepted and used within a particular industry or profession.</p> <p>The formula, <math>x + 1 = 7</math> where <math>x</math> stands for . . .</p>
Monospace font  (Looks like teletype or typewriter text)	<p>All programming identifiers and GUI elements. This convention includes:</p> <ul style="list-style-type: none"> <li>The <i>names</i> of directories, files, folders, configuration objects, paths, scripts, dialog boxes, options, fields, text and list boxes, operational modes, all buttons (including radio buttons), check boxes, commands, tabs, CTI events, and error messages.</li> <li>The values of options.</li> <li>Logical arguments and command syntax.</li> <li>Code samples.</li> </ul> <p>Also used for any text that users must manually enter during a configuration or installation procedure, or on a command line.</p>	<p>Select the <code>Show variables on screen</code> check box.</p> <p>In the <code>Operand</code> text box, enter your formula.</p> <p>Click <code>OK</code> to exit the <code>Properties</code> dialog box.</p> <p>T-Server distributes the error messages in <code>EventError</code> events.</p> <p>If you select <code>true</code> for the <code>inbound-bsns-calls</code> option, all established inbound calls on a local agent are considered business calls.</p> <p>Enter <code>exit</code> on the command line.</p>
Square brackets ([ ])	A particular parameter or value that is optional within a logical argument, a command, or some programming syntax. That is, the presence of the parameter or value is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information.	<code>smcp_server -host [/flags]</code>
Angle brackets (<>)	<p>A placeholder for a value that the user must specify. This might be a DN or a port number specific to your enterprise.</p> <p><b>Note:</b> In some cases, angle brackets are required characters in code syntax (for example, in XML schemas). In these cases, italic text is used for placeholder values.</p>	<code>smcp_server -host &lt;confighost&gt;</code>





# Index

## Symbols

__DEFAULT_	637
.rbn file	43
.zcf file	48
[] (square brackets)	825
< > (angle brackets)	825

## Numerics

800 numbers	482
-------------	-----

## A

abandoned call function	527
Access Code for switch	616, 641, 679, 705
access control	418
Access Numbers	477, 595, 678, 679, 705
ACDQueue	359, 451
acfdata function	559
Acknowledgement object	81, 156
Acknowledgement object button	79
ActiveServerName function	505
Add Record object	81
Add Record object button	80
ADDP options	701
agent	
best fit skills	338
capacity	676
groups logged on statistic	579
last routed	237
loading statistic	722
multiple extensions	675
not ready	508
ready statistics	515
reporting on	661
reports	749
reservation	654
statistics	579
targets and stat server	361

Agent Capacity distribution model	474, 479, 508, 676, 724
agent group target type	359
agent target type	359
agent_reservation option	629, 673
Alarm function	506
Alarm-level logs	752
angle brackets	825
ANI	
expressions	91
function	451
object	81, 374
strings in Lists	73, 482
ANI object button	78
AnswerCall function	506
Answering objective	144
applying Stat Server to strategies	347
Aspect switch and skills routing	653
Assign object	81, 120
Assign object button	79
async	
Custom Server	697
async option for Custom Server	697
Attach Categories object	81
Attach Categories object button	79
Attach function	451
attached data	
Autoresponse object	189
extrouter_timeout option	642
hide_private_data option	643
Interaction Data	72
Peg function	557
SendEvent function	533
Update function	467
UpdateBusinessData function	468
UpdateInteractionData function	469
use_ivr_info option	680
attaching several key-value pairs	467
AttributeReason	655
Attributes	375
AttributeUserData	658, 661, 663

audience	
defining	21
autobot	177
automatic response	162, 165, 176, 181, 182, 184
automatic_attach option	632
Autoresponse object	81, 156
Autoresponse object button	79
<b>B</b>	
backup T-Server	458
backup URS	
AgentStates	555
becomes primary	649
pickup calls	650
backup_mode option	703
backup_priority_level option	703
backupserver option	703
Basic Authentication access, Web Service	780
BearerCapability function	449, 453
Best Fit	333, 338
binary subroutine parameters	68
BinarySecurityTokenPEM	784
BlockDN function	589
brackets	
angle	825
square	825
buffer_size option for Custom Server	697
business attributes	102, 143, 453, 454, 461, 466
Business object	81, 418
Business object button	78
Business Process	149
definition	98
Business Rules	375
BusinessData function	453
BusinessDataNT function	454
business-priority routing	821
Busy	
DNs	554, 730
icon	77
IVR	594
object	81, 407
StatAgentLoading	722
StatAgentOccupancy	723
statistic	579
transition_time	672
treatment object	407
treatment reset	442
treatment settings	598
treatment timeouts	367
treatment_delay_time	673
Busy treatments	357, 366, 405
when not applied	370
bytecode option	695

## C

call forward on no answer	606
call parking option	680
Call Strategy	121
parameters	67
Call Strategy object button	79
Call Subroutine object	81, 121
call_monitoring option	634
call_tracking option	633, 634
calllage function	559
caller-entered-digits	455
CallID function	454
CallInfo functions	
ACDQ	451
ANI	451
Attach	451
BearerCapability	453
BusinessData	453
BusinessDataNT	454
CallID	454
CallType	454
CallUUID	455
CED	455
ConnID	455
DeleteAttachedData	456
Dest	456
DNIS	456
GetCurrentLeg	457
GetCurrentSwitch	458
GetCurrentTServer	458
GetCustomerSegment	459
GetMediaType	459
GetRoutingPoint	460
GetServiceObjective	460
GetServiceType	460
InformationDigits	461
InteractionData	461
InteractionDataINT	462
LATA	462
NPA	462
NPANXX	463
Orig	463
PACCCode	463
PACType	463
RequestType	464
StateCode	465
UData	465
UDataINT	466
Update	467
UpdateBusinessData	468
UpdateInteractionData	469
Callinfo functions	
ExtensionUpdate	457
FirstHomeLocation	457
SetHomeLocation	464

CallParked	670
calls answered statistic	579
calls completed statistic	579
CallsDistributed function	560
CallsEntered function	560
CallsWaiting function	561
CallsWaiting predefined statistic	719
CallsWaiting statistic	575
CallType function	454
CallUUID function	455
campaign target type	359
Cancel call	77
Cancel call object	81
Cancel call treatment objects	407
Cat function	582
CC Analyzer	
as part of the Reporting Layer	744
CC Pulse	
as part of the Reporting Layer	742
CCTExtractTargets	590
CED function	455
change access	418
change_tenant option	635
changing parameters in subroutines	69
changing strategies	59
chat transcript	157
Chat Transcript object	81
Chat Transcript object button	79
Check Integrity tool	34, 36
check_call_legs option	635
CheckAgentsState function	507
CIM Platform	20
ClaimAgentsFromOCS function	508
classification segmentation	
object	377
Classification Server	157
Classify (segmentation) object	82
Classify object	82, 198, 377
Classify object button	79
Classify Segmentation object button	78
cleanup of stacked calls	633
clear target	330
Clear Targets check box	364
ClearTargets function	510
ClearThresholds function	562
ClearUpdateTrigger function	511
Clickatell	221
client-designated external routing	595
close_statistic_time option	703
close_unused_statistic option	703
Collect digits	77
Collect digits object	82
Collect digits treatment object	407
Comment	81
Comment object	82
commenting on this document	21
compat_treatments option	406, 636
configuration objects	64
configuration options for IRD	
by function	
bytecode	695
inactivity-timeout	695
user timeout	644
by name	
inactivity-timeout	644
configuration options for URS	
by function	
absence of routing object	638
ADDP	701
agent_att	629, 632
backup hierarchy	703
call tenant validation	683, 635, 672, 664, 643, 640, 644
DN type, default	677
enabling log output	684
External Router	641, 678, 679
hide attached data	643
Interaction Server option	646, 634, 680
KPL	701, 703
load balancing	645
management client connection port	645, 646
Message Server	683
monitoring_time	646
null value	647
parsing option	647, 653, 651, 652
reconnection to server	653, 671, 654, 655, 703, 655, 658, 660, 703, 641
RingBack treatment, default	642, 643
routing error behavior	648, 666
service_timeout	666, 632, 667
Stat Server, default	639
strategy	668, 638, 633, 682, 635
target precedence	668, 629, 664, 672, 663, 634, 671
T-Server errors	644
UNIX host name	673
unloaded CDN	673, 674
verifying agent availability	686, 687
by name	
agent_att	629
configuration options for URS by function	
multiple backup servers	631
.	632
backup_mode	632, 703
call_kpl_time	633, 634, 635, 703, 636, 637
default_db_server	637, 638, 639

emergency_verbose . . . . .	639, 640, 641
function_compatibility . . . . .	642
give_treatment . . . . .	642
hanged_call_time . . . . .	643
ignore_customer_id . . . . .	644
joint_work . . . . .	703
kpl-interval . . . . .	703
lds . . . . .	645, 703
management_port . . . . .	645, 646
new_parsing . . . . .	647
on_primary_changed . . . . .	647, 648, 649
pickup_calls . . . . .	650, 651, 652
reconnect_time . . . . .	653, 654, 655, 703, 655, 658, 660, 663, 664, 666
service_timeout . . . . .	666, 667, 668, 703
targets_order . . . . .	668, 669
Threshold . . . . .	669, 670
transfer_time . . . . .	671, 672
unix-socket-path . . . . .	673, 674, 676, 677, 678, 679, 680, 681, 682, 683
validate . . . . .	683, 684, 686, 687
Configuration Options functions	
ExcludeAgents . . . . .	470
GetConfigOption . . . . .	472
GetMediaTypeName . . . . .	474
Router . . . . .	474
SetCallOption . . . . .	475
SetDelayedAttach . . . . .	545
SetDNIS . . . . .	475
SetDNISOverride . . . . .	475
SetTranslationOverride . . . . .	476
SetTreatmentMode . . . . .	617
UseActivityType . . . . .	477
UseCustomAgentType . . . . .	478
UseCustomDNTType . . . . .	478
UseCustomPlaceType . . . . .	478
UseDNTType . . . . .	479
UseMediaType . . . . .	479
Configuring	
functions in IRD . . . . .	423
IRD objects . . . . .	63
options . . . . .	619
URS for use with Web services . . . . .	687
ConnID function . . . . .	455
consult calls . . . . .	665
Contact Server . . . . .	158
Content Analysis . . . . .	177
conventions	
in document . . . . .	824
type styles . . . . .	825
coordinated universal time . . . . .	487
cost-based routing . . . . .	555, 615, 722, 821
pseudo statistics . . . . .	736
cost-based routing option . . . . .	655, 661
costs-based routing solution . . . . .	727
count_calls option . . . . .	637
counters as pegs . . . . .	711
CountSkillInGroup function . . . . .	511
CountSkillInGroupEx function . . . . .	511
cpu_emergency_level option . . . . .	637
Create Email Out object . . . . .	157, 212
Create Email Out object button . . . . .	79
Create Interaction object . . . . .	82
Create Interaction object button . . . . .	79
Create Notification object . . . . .	82, 157
Create Notification object button . . . . .	79
Create SMS object . . . . .	82, 158
Create SMS object button . . . . .	79
Create SMS Out object . . . . .	82, 386
Create SMS Out object button . . . . .	81
CreateEmailOut object . . . . .	82
CreateNotification object . . . . .	217
CreateSkillGroup function . . . . .	365, 514
CreateSMS object . . . . .	221
creating strategies . . . . .	32
CurrentAgentAssignment statistic . . . . .	362, 640, 722
Custom Server . . . . .	107, 697
Custom Server options	
async . . . . .	697
buffer_size . . . . .	697
frequency . . . . .	698
hide_private_data . . . . .	698
life_time . . . . .	698
new_parsing . . . . .	698
Custom Variables . . . . .	166, 167, 185, 213, 218, 223
Customer Segment . . . . .	143, 436, 459
function . . . . .	436
Customer Segment, Multi Attach object . . . . .	141
Customer Server . . . . .	114
<b>D</b>	
DAP . . . . .	112
Data Access Point	
default routing . . . . .	730
Data Manipulation functions	
ExtensionAttach . . . . .	456
ExtensionData . . . . .	457
FindServiceObjective . . . . .	480
ListGetDataCfg . . . . .	482
database	
case sensitivity . . . . .	113
DAP . . . . .	112
DB Server . . . . .	112
fields . . . . .	113
queries . . . . .	112
SQL select statement . . . . .	107
storing strategy RBN file . . . . .	61
tables . . . . .	113
Database Access Point . . . . .	112

Database non-SQL	
Custom Server	114
stored procedures	112
Database Wizard	108
Database Wizard object	82, 107
Database Wizard object button	77
Date function	
double-byte characters	486
date function	
internationalization	486
Date object	82, 380
Date object button	78
date segmentation	380
double-byte characters	385, 407
object	380
Date/Time functions	
Date	486
DateInZone	486
Day	486
DayInZone	487
GetUTC	487
IsSpecialDay	488
IsSpecialDayEx	488
Time	489
TimeDifference	489
TimeInZone	490
TimeStamp	490
UTCAdd	490
UTCFromString	491
UTCToString	492
DateInZone function	486
Day function	486
Day of the Week object	82, 381
Day of the Week object button	78
DayInZone function	487
DB Server	112
debugging strategies	36
Debug-level logs	753
decision data for routing	661
default destination in Busy tab	343
default destination option	316, 638
Default function	
retired	617
See Default object	
Default object	82, 316
Default object button	78
default routed calls	494, 638, 655, 730
default storage location by tenant	61
default target selection	668
default_db_server option	637
default_stat_server option	361, 639
defaultignore	673
defining object parameters	81
definition of Business Process	98
Delay function	515
Delete user announcement	77
Delete User Announcement object	82
Delete user announcement treatment	409
DeleteAttachedData function	456
delimiters in target lists	138
DelimitTargetList macro	138
DeliverCall function	591
DeliverToIVR	591, 593
DeliverToIVR function	591, 593
Dest function	456
Destination Label target type	359
developing strategies	30
Digest Authentication access, Web Service	781
Distribute Custom Event object	82, 158, 226
Distribute Custom Event object button	79
DistributedPercentage function	562
DistributedWaitingTime function	563
DN target type	589
DN verification	684
DNIS	430
DNIS function	456
DNIS Lists	73, 482
DNIS object	82, 381
DNIS object button	78
Do Not Call object	82
Do Not Call object button	80
document	
change history	22
conventions	824
errors, commenting on	21
version number	824
documentation	
eServices User's Guide	822
Events and Models Reference Manual	822
IRD Help	822
Log Events Help	822
Orchestration Server Deployment Guide	822
Proactive Routing Solution Guide	822
Universal Routing Business Process User's Guide	821
Universal Routing Cost-Based Routing Configuration Guide	822
Universal Routing Deployment Guide	821
Universal Routing Routing Application Configuration Guide	821
Universal Routing Strategy Samples	821
double-byte characters	
Date function	486
date segmentation	385, 407
duration of busy treatment	407

## E

E-mail Accounts	170, 172, 195, 234, 237, 251, 258, 274
e-mail classification	177
E-mail objects	

Acknowledgement	161
AutoResponse	181
Chat Transcript	191
E-mail Server Java	157, 158, 160
E-Mail-to-SMS gateway	158, 221
emergency strategy	527
emergency_verbose options	639
endpoints	150
Entry object	82, 126
Entry object button	79
enumerators	102, 143
environment option	639
Error (Fault) Code Tables	118, 119, 125, 174, 187, 196, 203, 210, 215, 219, 224, 228, 233, 235, 240, 250, 253, 258, 261, 269, 272, 277, 280, 283, 289, 389, 392, 394, 396, 399, 402
error function	527
Error Handling	114
Error Segmentation	123
error segmentation codes	125
Error segmentation object	82
Error segmentation object button	79
eServices definition	20
eServices Open Media Interaction Models	822
eServices routing point	651
estimated waiting time	317, 721
event_arrive option	640
Event3rdServerFault message	152
Event3rdServerResponse message	152
EventReleased	671
EventRouteRequest function	430, 532, 550, 591, 644, 677, 729
EventRouteUsed	671
EventStillInQueue	633
EventTreatmentEnd function	409
ExcludeAgents function	470
Exist object button	79
Exit object	82, 127
ExpandGroup function	515
ExpandWFActivity function	517
export.log file	49
exporting strategies	46
expression and Function object	127
Expression Builder	78, 94, 383
expressions (logical)	87
expressions for segmentation	383
expressions, mathematical symbols	91
ExtensionAttach function	456
ExtensionData function	457
ExtensionUpdate function	457
external routing	477, 557, 595, 616, 641, 678, 679, 705
External Service object	82
External Services object button	77
extrouter_dn option	641
extrouter_timeout option	641
ExtrouterError function	518
ExtrouterStatus function	518
<b>F</b>	
Fast busy	77, 409
Fast busy object	82
Fast busy treatment object	409
fault codes	
Acknowledgement object	173
Autoresponse object	187
Chat Transcript object	196
Classify object	203
Create Interaction object	210
Create SMS Out object	389
CreateEmailOut object	215, 219
CreateSMS object	224
External Service object	118, 119
Force Logout object	394
Forward E-Mail object	232, 235, 258, 288
Identify Contact object	228, 240, 280
Redirect E-Mail object	253
Reply E-Mail from External Resource object	261
Screen object	268
Send E-Mail object	272
Send SMS Out object	392
Set Agent DND State object	396
Set Agent Media State object	399
Set Strategy State object	402
Stop Interaction object	276
Update Contact object	283
fields (database)	113
Find Interaction object	82, 158, 228
Find Interaction object button	79
FindServiceObjective function	435, 480
FirstHomeLocation function	457
font styles	
italic	825
monospace	825
Force functions	
Force	494
TRoute	493
Force Logout object	82, 393
Force Logout object button	80
Force object button	78
Force Routing object	82, 316
forward e-mail	158
Forward E-Mail object	83, 233
Forward E-Mail object button	79
frequency option for Custom Server	698
full control	418
function in expression	90
Function object	83, 127
Function object button	79

function\_compatibility option . . . . . 642  
 functions  
     by name  
         ACDQ . . . . . 451  
         acfgdata . . . . . 559  
         ActiveServerName .505, 506, 451, 506, 451  
         BearerCapability . . . . . 453, 589, 453, 454  
         callage . . . . . 559  
         CallID . . 454, 560, 561, 454, 455, 582, 590,  
             455, 507, 508, 510, 562, 511, 455, 511, 514  
         Date .486, 487, 617, 515, 456, 591, 593, 456,  
             562, 563, 456  
         ExcludeAgents470, 515, 517, 456, 457, 518  
         FindServiceObjective . . 480, 457, 493, 494  
         GetAvgStatData . . 564, 472, 457, 458, 459,  
             495, 617, 518, 564, 495, 459, 474, 565,  
             495, 473, 519, 496, 594, 460, 519, 495,  
             487, 617  
         Increment617, 596, 461, 462, 617, 565, 488  
         JumpToStrategy . . . . . 522, 523  
         KeepQueue . . . . . 597, 497, 498  
         LATA . . . . . 462  
         lcfgdata . . . . . 559  
         ListGetDataCfg. . 482, 498, 499, 483, 484  
         Multiskill . . . . . 526  
         NotDistributedPercentage . . 566, 567, 462,  
             463  
         OnCallAbandoned . . . . . 527, 617, 527, 463  
         PACCCode . . 463, 557, 568, 528, 602, 603  
         Rand . . . . . 529, 464, 530, 569, 607, 474  
         SData . . . . . 572  
         sdata . . . . . 559  
         SelectDN 608, 530, 532, 534, 544, 475, 545,  
             475, 464, 500, 545, 546, 476, 548, 574,  
             575, 500, 612, 577, 617, 476, 617, 549,  
             613, 618, 465, 582, 583, 584, 585, 586,  
             587, 588, 614, 550, 615  
         TargetComponentSelected 551, 552, 489,  
             490, 553, 490, 615, 493  
         UData . . 465, 466, 554, 467, 468, 469, 554,  
             477, 554, 555, 579, 478, 479, 490, 491,  
             492  
         VQSelected . . . . . 555  
     functions list . . . . . 430

## G

Generic segmentation object . . . . . 83, 383  
 Generic segmentation object button . . . . . 78  
 Genesys Outbound Contact . . . . . 294  
 get functions . . . . . 452  
 GetAvgStatData function . . . . . 564

GetConfigOption function . . . . . 472  
 GetCurrentLeg function . . . . . 457  
 GetCurrentSwitch function . . . . . 458  
 GetCurrentTServer function . . . . . 458  
 GetCustomerSegment function . . . . . 459  
 GetIntegerKey function . . . . . 495  
 GetLastError  
     See Error Segmentation  
 GetLastError function  
     retired . . . . . 617  
 GetLastErrorInfo function . . . . . 518  
 GetMaxStatData function . . . . . 564  
 GetMaxSubList function . . . . . 495  
 GetMediaType function . . . . . 459  
 GetMediaTypeName function . . . . . 474  
 GetMinStatData function . . . . . 565  
 GetMinSubList function . . . . . 495  
 GetObjectProperty function . . . . . 473  
 GetPriority function . . . . . 519  
 GetRawAttribute function . . . . . 496  
 GetRemoteAccessCode function . . . . . 594  
 GetRoutingPoint function . . . . . 460  
 GetServiceObjective function . . . . . 460  
 GetServiceType function . . . . . 460  
 GetSkillInGroupEX . . . . . 520  
 GetSkillInGroupEx . . . . . 522  
 GetSkillInGroupEx function . . . . . 519  
 GetStringKey . . . . . 495  
 GetStringKey function . . . . . 495  
 GetUTC function . . . . . 487  
 give\_treatment option . . . . . 642  
 Global Interaction Type . . . . . 466  
 GMT . . . . . 487  
 Greenwich mean time . . . . . 487

## H

ha\_option . . . . . 651  
 hanged\_call\_time option . . . . . 643  
 help file  
     Interaction Routing Designer Help . . . . . 822  
 hide\_private\_data option . . . . . 643  
 hide\_private\_data option for Custom Server 698  
 hiding data in log . . . . . 644  
 high availability option . . . . . 621  
 Hot Standby . . . . . 506, 651  
 HTTP Authentication . . . . . 779  
 HTTP bridge . . . . . 621, 687  
 HTTP bridge options . . . . . 772  
 HTTP bridge verbose level . . . . . 685  
 HTTP Interface . . . . . 692  
 http\_conn\_idle\_timeout . . . . . 624  
 http\_log\_file option . . . . . 624  
 http\_log\_size option . . . . . 624  
 http\_port . . . . . 356  
 HTTP/HTTPS . . . . . 772

Identify Contact object	83, 158, 237
Identify Contact object button	79
If object	83, 129
If object button	79
ignore_customer_id option	644
import.log file	51
importing strategies	46
importstr folder	59, 61
inactivity-timeout IRD option	644, 695
IncrementPriority function	596
IncrementPriorityEx function	596
InformationDigits function	449, 461
INSERT command in query	113
Inter Server Call Control	557, 595, 641, 678
interaction	
attributes	102, 143, 657
queue definition	150
Interaction Concentrator (ICON)	742, 745
Interaction Data function	461
Interaction DataINT function	462
Interaction Design window	149
Interaction processing	
Screen	159
interaction processing	
acknowledgement	156
autoresponse	156
chat transcript	157
classify	157
Create Email Out	157
Create Notification	157
Create SMS	158
Distribute Custom Event	158
Find Interaction	158
forward e-mail	158
Identify Contact	158
Queue Interaction	159
queue name	151
redirect e-mail	159
Render Message Content	159
Route Interaction	159
screen	158
send e-mail	160
Submit New Interaction	160
Update Contact	160
Update Interaction	160
Update UCS Record	160
Workbin	160
Interaction Routing object button	78
Interaction SDK	115
Interaction Server	149, 150, 151, 667, 699
Interaction-level logs	752
interactions	
waiting function	572
Interactive Insights reports	749

internationalization	
date function	486
supported formats	380
Date function	486
time segmentation	385, 407
Internet Contact Solution	
Acknowledgement	161
Autoresponse	181
Interruptible check box	411
inv_connid_errors option	644
InVQWaitTime function	565
InVQWaitTime predefined statistic	719
InVQWaitTime statistic	575
IRD objects	
Data & Services	
Database	108
External Service	115
Web Service	119
E-mail objects	
Acknowledgement	161, 181
Chat Transcript	191
Miscellaneous	
Assign	120
Error segmentation	123, 127
Function	127
If	129
Macro	130, 140, 141, 146
Multimedia	
Acknowledgement	161, 178, 181
Chat Transcript	191, 198, 212, 205, 217, 221
Distribute Custom Event	226
Find Interaction	228, 233
Identify Contact	237
Multiscreen	241
Redirect Email	250, 254, 259
Screen	263, 269, 274, 277
Update Contact	281, 285, 290
Outbound	
Add Record	295
Do Not Call	299
Processed	302
Reschedule	305
Update Record	307
Routing	
Default	316
Load Balancing	317
Percentage	318
Selection	326, 333, 346, 347
Workforce	353
Segmentation	
ANI	374
Business	418
Classification	377

Date	380, 381
Generic	383
Time	385, 407
SMS	
Create SMS Out	386
Send SMS Out	390
Treatment	409, 411
Busy	407
Cancel call	407
Delete user announcement	409
IVR	409
music	410
Play Announcement	411, 412
RAN	413, 414, 415
Set Default Destination	415
Text to Speech	415, 416
Verify Digits	416
Workflow and Resource Management	
Force Logout	393
Set Agent DND State	395, 398, 400
IRD timeout	695
IRD working directory	59
ISCC	557, 595, 641, 678, 679, 705
IsSpecialDay function	488
IsSpecialDayEx function	488
italics	825
IVR	77
IVR deliver to function	593
IVR object	83
IVR port limit	681
IVR treatment object	409
IVR wait time announcement	562, 566, 569, 576

## J

joint_work option	703
JumpToStrategy function	522
JumpToTenant function	523

## K

Keep Alive Protocol	701
KeepQueue function	597
key-value	428, 451, 453, 467
Knowledge Manager	158, 159
creating messages for SMS	221
kpl-interval option	703
kprl_priority option	703
KVListGetKey function	497
KVListGetListValue function	498
KVListGetSize function	498
KVListGetStringValue function	498

## L

Last Agent Routing	114
Last Called Agent information	237
Last Routed Agent	237
LastCalledAgent_EmployeeID	241
LATA function	449, 462
LBR_DEST key in User Data	658
lcfgdata function	559
lds option	645
licensing for high availability	651
life_time	
Custom Server	698
life_time option for Custom Server	698
limitation for key-value strings	453
limitations when developing a strategy	30
List Manipulation	473
List Manipulation functions	
GetIntegerKey	495
GetMaxSubList	495
GetMinSubList	495, 496
GetStringKey	495
KVListGetKey	497
KVListGetListValue	498
KVListGetSize	498
KVListGetStringValue	498
ListGetInteger	498
ListGetKey	499
ListGetSize	499
ListGetString	499
ListLookup	483
ListLookupCfg	484
SetIntegerKey	500
SetObjectProperty	476
SetRunTimeMode	548
SetStringKey	500
ListGetDataCfg function	482
ListGetInteger function	498
ListGetKey function	499
ListGetSize function	499
ListGetString function	499
ListLookup function	483
ListLookupCfg function	484
Lists	73
load balancing and event_arrive option	640
Load Balancing object	317
Load balancing object	83
Load Balancing object button	78
Load-balancing statistic	580, 658
loading	
non-voice strategy	
see Business Process User's Guide	100
strategy	65
voice strategy	38
Local Access Transport Area	462
local variables	97

log events	621
log file, verbose option	684
log options	703
log when no high availability license	651
log_buffering option	703
log_file_name option	703
log_file_size option	703
log_remove_old_files option	703
log, hiding private data	644
Logging during irregular operation	753
logging options	701
logical expressions	87
symbols	90
syntax	92
logs	
Management Layer	822
lost interactions	671

<b>M</b>	
Macro object	83, 130
Macro object button	79
macros	74
macros for URS	
by name	
checkbusinessrules	138
delimittargetlist	138
ring-no-answer	138
Management Layer logging	822
management_port option	645
mandatory treatments	405
mathematical symbols in expressions	91
max_cpu_objects_slice option	646
Maximum value of statistic	325, 332, 352
max-submitted-interactions option	646
MCR	149
media type	102, 130, 474, 724
message log	621
Message Server	620, 645, 646
Message Server options	701
message types for e-mail processing	151
Minimum value of statistic	325, 332, 352
Miscellaneous functions	
SetLastError	546
SetQueueLabel	548
miscellaneous functions	
ActiveServerName	505
Alarm	506
AnswerCall	506
CheckAgentsState	507
ClaimAgentsFromOCS	508
ClearTargets	510
ClearUpdateTrigger	511
CountSkillInGroup	511
CreateSkillGroup	514

Delay	515
ExpandGroup	515
ExpandWFActivity	517
ExtrouterError	518
ExtrouterStatus	518
GetLastErrorInfo	518
GetPriority	519
JumpToStrategy	522
JumpToTenant	523
MultiSkill	526
OnCallAbandoned	527
OnRouteError	527
Print	528
Rand	529
ReleaseCall	529
ResetBusyTreatments	530
SelectTargets	530
SendEvent	532
SendRequest	534
ServerStatus	544
SetInteractionAge	545
SetUpdateTrigger	549
SuspendForEvent	550
TargetComponentSelected	551
TargetObjectSelected	552
TargetSelected	552
Timeout	553
UDataINT	554
UpdateScript	554
UseAgentState	554
UseAgentStatistics	555
VQSelected	555
Miscellaneous objects	78
Assign	120
Call Strategy	121
Entry	126
Error segmentation	123
Exit	127
Function	127
If	129
Macro	130
Multi Assign	140
Multi Attach	141
Multi Function	146
modifying strategies	59
monitoring calls	65, 362
monitoring DNs	623
monitoring options	701
Monitoring View	39, 65
monitoring_time option	646
monospace font	825
MS-Tserver replacement	699
Multi Assign object	140
Multi Attach object	141
Multi Function object	146
MultiAssign	83

MultiAssign object button	79
MultiAttach object	83
MultiAttach object button	79
Multi-Channel Routing	149
MultiFunction	146
MultiFunction object button	79
Multimedia objects	79
Classify	198
Create Email Out	212
Create Interaction	205
CreateEMailOut	212
CreateNotification	217
CreateSMS	221
Distribute Custom Event	226
Find Interaction	228
Forward E-mail	233
Identify Contact	237
Redirect E-Mail	250
Render Message Content	254
Reply E-mail from External Resource	258
Screen	263
Send E-mail	269
Stop Interaction	274
Submit New Interaction	277
Update Contact	281
Update Interaction	285
Update UCS Record	290
multiple DNs	675
MultiScreen object	83
Multi-Screen object button	79
Multi-Site support	595
Multiskill function	526
Music	77
Music object	83
Music treatment object	410

## N

network routing	477, 616, 641, 679, 705
Network Routing Solution	
network T-Server functions	449
Universal Routing	19
new_parsing option	647
new_parsing option for Custom Server	698
no access	418
Non-configured DN target type	360
Non-monitored DN target type	360
not ready agent	508
NotDistributedPercentage function	566
NotDistributedWaitingTime function	567
NPA function	462
NPANXX function	463
null_value option	647
number of interactions waiting	572
Number translation	682, 705

## O

object	
access control	418
name	64
permission	418
properties	85
object browser	100
object types	
Data and Services	107
Function	127
Miscellaneous	120
Multimedia	149
Routing	311
Segmentation	374
SMS	386
Voice Treatment	404
Workflow and Resource Management	392
Workforce	353
Objective Table	144, 480
objects	
Acknowledgement	161
ANI	374
Assign	120
Autoresponse	181
Business	375
Busy	407
Call Subroutine	121
Cancel Call	407
Chat Transcript	191
Classify	377
Classify Segmentation	377
Collect Digits	407
Database Wizard	107
Date	380
Day of Week	381
Default Routing	316
Delete User Announce	409
DNIS	381
Entry	126
Error Segmentation	123
Exit	127
Fast Busy	409
Force Routing	316
Force routing	316
Function	127
Generic	383
If	129
IVR	409
Load Balancing	317
Macro	130
Multi Assign	140
Multi Attach	141
Multi Function	146
Music	410
Pause	411

Percentage	318	close_unused_statistic	703
Percentage routing	318	compat_treatments	636
Play Announcement	411	count_calls	637
Play Announcement Coll Digits	412	cpu_emergency_level	637
Play Application	412	DB Interaction Server option	646
Queue Interaction	320	default_db_server	637
RAN	413	default_destination	638
Record User Announcement	414	default_stat_server	639
ringback	415	environment	639
Route Interaction	322	event_arrive	640
Selection	326	extrouter_dn	641
Service Level	333	extrouter_timeout	641
Set Default Destination	415	function_compatibility	642
Silence	415	give_treatment	642
Statistics	346	hanged_call_time	643
Switch-to-Strategy	347	hide_private_data	643
Text to Speech	415	ignore_customer_id	644
Text to Speech Coll Digits	416	inv_connid_errors	644
Time	385, 407	lds	645
Verify Digits	416	management_port	645
Workbin	348	max_cpu_objects_slice option	646
Workforce	353	monitoring_time option	646
on_primary_changed option	647	null_value	647
on_route_error option	648	on_primary_changed	647
on_router_activated option	649	on_route_error	648
OnCallAbandoned function	527	on_router_activated	649
OnError function		parsing option	647
retired	617	pickup_calls	650
See Error Segmentation		prestrategy	651
OnRouteError function	527	pulse_time	652
Open Media	115	reconnect_time	653
option	631	reduced	653
option override	475	reg_attempts	654
options	629, 632, 643, 653, 667, 668, 674	reg_delay	655
ADDP	701	reg_mode	703
Custom Server	697	report_reasons	655
Custom Server buffer size	697	report_statistics	658
options list	621	report_targets	660
options (Custom Server)		request_timeout	663
frequency	698	reservation_pulling_time	664
hide_private_data	698	route_consult_call	664
life_time	698	run_time_mode	666
new_parsing	698	service_timeout	666
options (Interaction Server)	699	skip_targets	667
options (IRD)		targets_order	668
bytecode	695	ThresholdDestination	669
inactivity-timeout	644, 695	ThresholdGroup	669
options (URS)		ThresholdSwitch	670
agent_reservation	629	transfer_time	671
automatic_attach	632	transfer_to_agent	671
call_kpl_time	633	transit_db	671
call_monitoring	634	transition_time	672
call_tracking	634	treatment_delay_time	672
change_tenant	635	unix_socket_path	673
check_call_legs	635	unloaded_cdn	673
close_statistic_time	703	use_agent_capacity	674

use_agentid	676
use_dn_type	677
use_extrouter	678
use_extrouting_type	679
use_ivr_info	680
use_parking_threshold	680
use_service_objective	681
use_translation	682
using	683
validate	683
verbose	684
verification_mode	686
verification_time	686
verification_time_agent	687
verification_time_dn	687
options (URS-retired)	
backup_mode	703
backup_priority_level	703
backupserver	703
close_statistic_time	703
close_unused_statistic	703
joint_work	703
kpl-interval	703
kpri_priority	703
log_buffering	703
log_file_name	703
log_file_size	703
log_remove_old_files	703
max_loading	703
strategy	703
Orig function	463
original agent routing	237
Outbound	
buttons	80
Outbound Contact	294
Outbound Contact Server	639, 722
Outbound strategy-building objects	294
override for server level options	475
P	
PACCode function	463
packaging a strategy	46
PACType function	449, 463
parameters in subroutines	67, 69
parking threshold mechanism	670, 681
parking threshold option	680
ParkingThreshold	669, 670
parsing single quotes	
Custom Server	698
Pause	77, 411
Pause object	83
Pause treatment object	411
performance enhancement	653
Peg function	557
Peg option	622
Pegs	
Automatic, defined	711
PegSF	557
PegValue function	557
Percentage	
for distribution	318
Percentage object	83, 318
Percentage object button	78
performance	475, 515, 519, 663
permissions for objects	418
pickup_calls option	650
place group target type	360
place in Queue	159
place target type	360
Play announcement	77
Play announcement and collect digits object	83
Play announcement object	83
Play Announcement treatment	411
Play Announcement/Collect digits treatment	412
Play application	77
Play application object	83
Play Application treatment object	412
Play recorded announcement	77
port limit in IVR	681
PositionInQueue function	568
PositionInQueue predefined statistic	719
PositionInQueue statistic	575
PostRoute	347
predefined statistics, using	716
predicted wait time	721
prediction in Busy tab	343
PreferredAgent_EmployeeID	241
prestrategy option	651
pre-written response	162, 165, 176, 181, 182, 184
primary T-Server	458
primary URS	
distributing interactions	649
pickup calls	650
Print function	528
Priority function	602
priority routing	
age	442
busy treatments	369
global priority function	437
increment priority	438
KeepQueue function	438
service level	345
service objective	442, 681
service objective threshold	145
set global priority	441
virtual queue	445
wait time	442
PriorityLimits function	603
PriorityTuning function	603
Proactive routing objects	294
Processed object	83

Processed object button	80
prohibited symbols	31
properties dialog box for objects	85
protocol for e-mail processing	151
pseudo statistics	736
pulse_time option	652
Push Preview mode	310

## Q

queries	112
queries in Database object	108
Queue	
ACD	359
definition	150
display name vs. Script name	151
IRD object	159
number of calls in statistic	579
reports	749
types	150
Queue Group	553
queue group target type	360, 578
Queue Interaction	159
Queue Interaction object	83, 320
Queue Interaction object button	78
quotation marks in variables	96
quote usage in functions	429
quotes in DB Wizard	113

## R

race condition	370
RAN	77
RAN object	83
RAN treatment object	413
Rand function	529
rbn file storage location	61
read access	418
ready for next interaction	628
Ready state statistic	579
Reason Code	160, 275
reconnect_time option	653
Record user announcement	77
Record User Announcement object	83
Record user announcement treatment	414
Recorded Announcement object	83
red X in IRD main window	34
redirect e-mail	159
Redirect E-Mail object	83, 250
Redirect E-Mail object button	79
reduced option	653
reg_attempts option	654
reg_delay option	655
reg_mode option	703
reject-subsequent-request option	630

ReleaseCall function	529
Relevancy Threshold	189
Render Message Content object	83, 159, 254
Render Message Content object button	79
replace string function	585
Reply E-Mail from External Resource object	83
Reply E-mail from External Resource object	258
Reply E-Mail From External Resource object button	79
report_reasons option	655
report_statistics option	658
report_targets option	660
Reporting	556
location of templates	749
peg	557
service level	557
Reporting functions	
OnCallAbandoned	527
Peg	557
PegValue	557
Reporting Layer	
applications	742
CC Analyzer	744
CC Pulse	742
Interactive Insights	749
reporting using categories	157
request_timeout option	663
Request3rdServer message	152
request-collection-time option	630
Requested Skill	141
Requested Skill in Multi Attach object	142
RequestType function	449, 464
re-routing attempts	624
Reschedule object	83
Reschedule Record object button	80
reservation	629, 653, 672
reservation_pulling_time option	664
reserved symbols	31
ResetBusyTreatments function	530
ResetStatAdjustment	569
ResetStatAdjustment function	569
Response Manager	176
retired options	703
Ringback	77
Ringback object	84
Ringback treatment object	415
ring-no-answer	138, 536, 606
Route Interaction	159
Route Interaction object	84, 322
Route Interaction object button	78
route_consult_call option	664
RouteCall	607
RouteCall function	607
Routed function	607
Router function	474
Router High Availability Mode turned off	651

Router Self-Awareness	317
router_ha_option	621, 651
routing	
busy treatments	357
conditional	146
decision data	661
Default object	316
Force Routing object	316
General selection	326
Load Balancing	317
loading a strategy	38
non-Genesys servers	115
object busy treatment	366
Percentage	318
post	357
reporting decisions	661
Route Interaction object	322
Service level	333
skill-based	522
Statistics	346
strategy samples	821
strategy storage location	61
Switch to strategy	347
target types	373
using Statistics	346
Workforce	353
Routing Design window	28
Routing object	78
routing point function	460
routing point target type	360
routing point, eServices	651
Routing rule, creation	312
routing rule, Workforce	353
routing to original agent	237
routing using categories	157
RStatCallsInQueue predefined statistic	719
RStatCallsInTransition	719, 726
RStatCallsInTransition predefined statistic	719
RStatCallsInTransitionEx	727
RStatCallsInTransitionEx predefined statistic	719
RStatCost predefined statistic	719, 727
RStatExpectedLoadBalance predefined statistic	719
RStatLBEWTLAA predefined statistic	719
RStatLoadBalance predefined statistic	719
RStatRoundRobin predefined statistic	719
RStatTimeInReadyStateMedia predefined statistic	719
run_time_mode option	666
<b>S</b>	
Same queue(_BACK_	321
Samples	821
saving a strategy	60
saving, default storage by tenant	61
Scheduler	72
Scope in Variable list dialog box	94
Screen	159
screen interaction	158
Screen object	84, 263
Screen object button	79
screen pops	72
Screen Segmentation object	84, 384
Screen Segmentation object button	78
ScreenRuleMatch	267
SData function	572
sdata function	559
security	418
security banner message	32
Segmentation	
Screen	384
segmentation	
ANI	374
Business	418
classification code	377
Date	380
Day of the Week	381
DNIS	381
Generic	383
Time	385, 407
Segmentation object	77
Select statement	108, 112
SelectDN function	608
Selection object	84, 326
Selection object button	78
SelectTargets function	530
Send E-Mail	160
Send E-Mail object	84, 269
Send E-Mail object button	79
Send SMS Out object	84, 390
Send SMS Out object button	81
send to queue	159
SendEvent function	532
SendRequest function	534
sensitive log data	644
Server Disconnected event	702
ServerStatus function	544
service factor	342
Service Level object	333
priority	345
service factor	342
Service level object	84
Service level object button	78
Service Objective	143, 144
function	435, 437
Multi Attach object	141
PriorityTuning	606
Service Type	143
Multi Attach object	141
service_timeout option	666
Set Agent DND State object	84, 395

Set Agent DND State object button	80
Set Agent Media State object	84, 398
Set Agent Media State object button	80
Set default destination	77, 84
Set Default Destination treatment	415
set functions	452
Set Multimedia Strategy State	84
Set Multimedia Strategy State object	84, 400
Set Multimedia Strategy State object button	80
SetCallOption function	475
SetDelayedAttach function	545
SetDNIS function	475
SetDNISOverride	475
SetHomeLocation function	464
SetIntegerKey function	500
SetInteractionAge function	545
SetLastError function	546
SetObjectProperty function	476
SetQueueLabel function	548
SetRunTimeModefunction	548
SetStatAdjustment function	574
SetStatUpdate	405
SetStatUpdate function	575
SetStringKey function	500
SetTargetThreshold function	146, 612
SetThreshold function	577, 617
SetTranslationOverride function	476
SetTreatmentMode function	617
SetUpdateTrigger function	549
SetVQPriority function	613
Share Agent by Service Level Agreement	146
Share Agents by Service Level Agreement	821
sharing strategies	42
different environment	43
same environment	43
Short Message Service	158, 221
Silence	77
Silence object	84
Silence treatment object	415
size limitation for key-value strings	453
skill	
Best Fit	338
Expression Builder	87
function	514, 520, 526
levels	338
number of agents with skill	511
objects in strategy	90
reporting	552
statistics	521
target type	360, 363
virtual group	512
virtual groups	516
skill expression	
as target	365
creating	88, 330
excluding an agent	366
limitation	31
Multiskill function	527
performance	654
predefined statistics	718
Route Interaction object	322
Selection routing object	330
Service Level Routing object	333
Skill Expression Builder	88
skill-based routing	522
Aspect and Spectrum switches	654
reduced option	653
Workforce Management	356
skip_targets option	667
SMS	158
buttons	80
creating standard responses for	221
SMS Gateway for Mdaemon	221
SMS objects	
Create SMS Out	386
Send SMS Out	390
sms2email.com	221
SOAP/XML	771
spaces in names	64, 113, 555
special access	418
Spectrum switch and skill-based routing	653
Spectrum switch and skills routing	654
SQL queries	112
square brackets	825
stacked calls	633
Standard level of logging	752
standard response	156, 162, 165, 176, 181, 182, 184
Stat Server	
agent targets	361
default	361
predefined statistics	715
URS issues	727
StatAgentLoading	515, 722
StatAgentLoading predefined statistic	721
StatAgentLoadingMedia predefined statistic	721
StatAgentOccupancy	723
StatAgentOccupancy predefined statistic	721
StatAgentsAvailable	579
StatAgentsAvailable predefined statistic	721
StatAgentsBusy	579
StatAgentsBusy predefined statistic	721
StatAgentsInQueueLogin predefined statistic	721
StatAgentsInQueueReady predefined statistic	721
StatAgentsTotal	579
StatAgentsTotal predefined statistic	720
StatAverageHandlingTime	579
StatCallsAnswered	579
StatCallsAnswered predefined statistic	720
StatCallsCompleted	579
StatCallsCompleted predefined statistic	720

StatCallsInQueue	579
StatCallsInQueue predefined statistic	720
StateCode function	449, 465
StatEstimatedWaitingTime predefined statistic	720
StatExpectedWaitingTime predefined statistic	720
statistic min/max value used in routing.	325, 332, 352
Statistical	566
Statistical Days	438, 531, 580
Statistical functions	
CallsDistributed	560
CallsEntered	560
CallsWaiting	561
ClearThresholds	562
DistributedPercentage	562
DistributedWaitingTime	563
GetAvgStatData	564
GetMinStatData	565
InVQWaitTime	565
NotDistributedWaitingTime	567
PositionInQueue	568
ResetStatAdjustment	569
SData	572
SetStatAdjustment	574
SetStatUpdate	575
SetThreshold	577, 617
UseCapacity	579
statistics	
impact of invalid statistics on URS	728
persistent	728
pseudo	736
returning value	572
Stat Server	728
that can be modeled	579
Statistics object	84, 346
Statistics object button	78
statistics, pseudo	736
StatLoadBalance	580
StatLoadBalance predefined statistic	720
StatSelectMax	609
StatSelectMin	609
StatServiceFactor predefined statistic	720
StatTimeInReadyState predefined statistic	720
Stop E-mail predefined queue	324, 350
Stop Interaction	160
Stop Interaction object	84, 274
Stop Interaction object button	79
Stop Processing predefined queue	324, 350
storage locations by tenant	61
stored procedures	108, 112
StrAsciiBreak function	582
StrAsciiTok	582
strategies	
applying Stat Server to	347
strategy	
.rbn file	43, 61
Call Strategy object	121
components	43
creating	32
debugger	36
developing	30
linked nodes	153
loading	38
loading a strategy	65
modifying	59
packaging	46
samples	821
saving	60
sharing	42
storage location	61
troubleshooting	34
unpackaging	50
variable-driven	33, 92
strategy option	703
strategy sharing	
.zcf file	48
export.log file	49
import.log file	51
rules	52
strategy viewer	100
strategy-linked nodes	153
StrChar function	583
StrGetChar function	584
string constant limitation	424
String Manipulation functions	
Cat	582
StrAsciiBreak	582
StrAsciiTok	582
StrGetChar	584
StrLen	584
StrNextTokInd	585
StrReplace	585
StrStr	585
StrSub	585
StrTargets	586
StrToLower	587
StrToUpper	588
string subroutine parameters	68
StrLen function	584
StrNextTokInd	585
StrReplace function	585
StrStr function	585
StrSub function	585
StrTargets function	586
StrToLower function	587
StrToUpper function	588
stuck calls	633
Submit New Interaction object	84, 160, 277
Submit New Interaction object button	79
subroutine	

Call Strategy	121
changing parameters	69
parameters	67
parameter formats	68
subroutine samples	424
SuspendForDN function	614
SuspendForEvent function	550
SuspendForTreatmentEnd function	615
Switch Access Code	477, 616, 641, 679, 705
Switch to Strategy object	84, 347
Switch to strategy object button	78
switching off features	653
switch-IR number translation	
configuration option for	682
symbols	31, 90
synchronous mode	
Custom Server	697

## T

Table Access	483
tables (database)	113
target	
clear	330
list processing	138
same agent, different stat server	361
types in routing objects	373
variables	92
target and busy treatment timeouts	367
Target Manipulation functions	
BlockDN	589
CCTExtractTargets	590
DeliverCall	591
DeliverToIVR	591, 593
GetRemoteAccessCode	594
IncrementPriority	596
IncrementPriorityEx	596
KeepQueue	597
Priority	602
PriorityLimits	603
PriorityTuning	603
RouteCall	607
Routed	607
SelectDN	608
SetTargetThreshold	612
SetVQPriority	613
SuspendForDN	614
SuspendForTreatmentEnd	615
Translate	615
target selection	317, 326
target type	
ACDQueue	359
agent	359
agent group	359
available types	578
campaign	359

Destination Label	359
non-configured DN	360
place	360
place group	360
queue group	360
routing point	360
skill	360
variable	360
TargetComponentSelected function	551
TargetObjectSelected function	552
targets_order option	668
TargetSelected function	552
tenant	
report_targets option	662
tenant storage locations for rbn	61
Text to speech	77
Text to speech and collect digits	77
Text to speech and collect digits object	84
Text to speech and Collect digits treatment	416
Text to speech object	84
Text to Speech treatment	415
TGetAccessNumber	595
third party server message types	152
Third-Party Server	115
Threshold parameter	578
threshold expression	327
threshold, relevancy	189
ThresholdDestination option	669
ThresholdGroup option	669
ThresholdSwitch option	670
Time function	489
Time object	84, 385, 407
Time object button	78
time segmentation	
internationalization	385, 407
time standard	487
time zone	385
TimeDifference function	489
TimeInZone function	490
Timeout function	553
timeout in IRD	695
timeout information in Busy tab	343
timeouts	
busy treatments	367
TimeStamp function	490
T-Library functions	
EventRouteRequest	430, 532, 550, 591, 644, 677, 729
EventTreatmentEnd	409
toll-free number Lists	73, 482
tools_tuneup option	116
Trace Mode	702
Trace On command	40
Trace View command	40
Trace-level logs	753
transfer_time option	671

transfer_to_agent option	671
transit_dn option	671
transition_time option	672
Translate	615
Translate function	615
transiton_time option	589
Treatment objects	
Busy	407
Cancel call	407
Collect digits	407
defined	418
Delete User Announcement	409
Fast busy	409
IVR	409
Music	410
Pause	411
Play Announcement	411
Play Announcement/Collect Digits	412
Play Application	412
RAN	413
Record user announcement	414
Ringback	415
Set Default Destination	415
Silence	415
Text to Speech	415
Text to Speech and Collect digits	416
Verify Digits	416
treatment_delay_time option	672
troubleshooting	528, 788
strategies	34
URS and Stat Server	727
TRoute function	493
T-Server errors	624
T-Server protocol for e-mails	151
turning off features	653
Type in Variable list dialog box	94
type styles	
conventions	825
italic	825
monospace	825
typographical styles	824, 825
<b>U</b>	
UData function	465
UDataINT function	466, 554
Universal Callback Server	545
Universal Contact Server	152, 160
Universal Contact Server Database	165, 184, 193
Universal Contact Server database	200
Universal Routing Server	
counters	711
efficiency	653
function compatibility	642
functions	423
number translation	705
objects	63
options	619
Stat Server dependencies	727
strategies	27
URS-Behind solution	797
unix_socket_path option	673
unloaded_cdn option	673
unpackaging a strategy	50
Update Contact	160
Update Contact object	84, 281
Update Contact object button	79
Update function	467
Update Interaction object	84, 160, 285
Update Interaction object button	79
Update record object	84
Update Record object button	80
Update UCS Record object	84, 160, 290
Update UCS Record object button	79
UpdateBusinessData function	468
UpdateInteractionData function	469
UpdateScript function	554
updating strategies	59
URS-Behind solution	797
use_agent_capacity option	674
use_agentid option	676
use_dn_type option	677
use_extrouter option	678
use_extouting_type option	679
use_ivr_info option	680
use_parking_threshold option	680
use_service_objective option	681
use_translation option	682
UseActivityType function	477
UseAgentState function	554
UseAgentStatistics function	555
UseCapacity function	579
UseCustomAgentType function	478
UseCustomDNTType function	478
UseCustomPlaceType function	478
UseDNTType function	479
UseMediaType function	479
User Data	207, 210
Attach function	451
attaching header fields	272
AttributeUserData example	152
BusinessData function	453
Chat Transcript object	193
GetCustomerSegment function	459
GetServiceObjective function	460
GetServiceType function	461
Interaction Concentrator	745
InteractionData function	461
InteractionDataINT function	462
LBR_DEST key	658
Proactive Routing	303
report_reasons option	656

report_statistics option	658
report_targets option	661, 663
SendEvent function	532
SendRequest function	536
UData function	465
User Data for e-mail	152
UsernameTokenDigest	784
UsernameTokenText	784
using option	683
UTC	487
UTCAdd function	490
UTCFromString function	491
UTCToString function	492
utility subroutines	424
<b>V</b>	
validate option	683
variable target type	360
variables	
assigning in Function object	127
assigning values	94
in a strategy	33, 92, 429
input and output	68
objects and expressions	92
pseudo statistics	736
quotes	96
routing rules	92
scope	97
specification	96
specifying in dialog boxes	430
type	97
verbose option	684
verification_mode option	686
verification_time option	686
verification_time_agent option	687
verification_time_dn option	687
verifications	653, 684
Verify digits	77
Verify digits object	85
Verify Digits treatment	416
version numbering, document	824
version of URS, function compatibility	642
virtual agent group	326, 330, 516, 716
virtual queue	
call restoration and stacked calls	633
DNIS	430
parameter for next event	633
routing objects	357
variable or string	357
virtual queue definition	150
virtual routing points	177, 360
Voice Callback	34, 545
Voice Treatment objects	77
VQSelected function	555

## W

wait time	317, 405, 442, 574, 597, 614, 615, 673
predicted	721
StatEstimatedWaitingTime	720
StatExpectedWaitingTime	720
virtual queue	438
wait time announcement	562, 566, 569, 576
waiting interactions	572
waiting time	673
Warm Standby	458, 506, 651
Web Service object	85, 119
Web Service object button	77
Web services	
configuring URS for use with	621, 687
Week segmentation object	82
Where clause	112
wizard	108
database	108
Workbin	160
Workbin object	85, 348
Workbin object button	78
Workflow and Resource Management	
buttons	80
Workflow and Resource Management objects	
Force Logout	393
Set Agent DND State	395
Set Agent Media State	398
Set Multimedia Strategy State	400
workflow view	149
Workforce object	85, 353
Workforce object button	78
working directory	59
Working directory for IRD	59
write access	418
<b>X</b>	
XML-based messages	772