



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

eServices Administrator's Guide

Genesys Engage Digital (eServices) 8.5.1

10/26/2023

Table of Contents

eServices Administrator's Guide	4
Interaction Server Administration	5
Interaction Server Limitations	7
Improving the Performance of the Interaction Server Database	9
General Remarks on Partitioning	10
Planning	11
Creating the Database	12
Carrying Out the Partitioning	13
Verification	15
Converting to and From BLOB	16
Event Logger	17
Deploying Event Logger	18
Managing Event Logger Data	19
Classification of Events in Event Logger	20
Event Logger Options	22
Using a Message Queue with Event Logger	25
Using the JMS Event Logger with Apache ActiveMQ	26
Using the Groovy Event Logger	29
Using the Groovy Event Logger with Apache Kafka	30
Using the Groovy Event Logger with Elasticsearch	32
E-mail Server Administration	34
JavaMail Properties	35
Delivery Status Notification and Message Disposition Notification	36
Customizing the Format of External Resource E-mails	37
CSS for HTML Chat Transcripts	38
UCS Administration	39
User Other than Schema Owner	43
Required Queries	44
Updating Interaction Data in a Routing Strategy	46
UCS Limitations	47
UCS Manager	48
Contact Identification	49
Contact Identification: Example	52
Specify Attributes To Check in Contact Identification	54
Customize Attribute Priority	55

Customize Contact Identification per Media Type	56
Contact Creation	58
Searching the UCS Database	59
Sizing for the UCS Database and Index	62
Configuring Analyzers for UCS Search	63
Search Syntax	66
Making an Attribute Searchable/Sortable from the Desktop	67
Chat Server Administration	69
Sizing Guide, Setting Load Limits, and Health Monitoring	70
Deploying a Chat Solution	71
Deploying High-Availability Chat Server	72
Configuring a secure connection to Cassandra	73
Initialization Cassandra scripts for Chat Server	74
Multilingual Processing in Chat Server	75
Masking Sensitive Data	76
Inactivity Monitoring	77
Matching Contact Attributes	78
How to send ESP requests to Chat Session from Workflow	79
Chat Server Reporting Statistics	80
Chat Server API selected notes and topics	81
Functional capabilities of chat protocol	82
File Transfer API for Agent Desktop	83
SMS Server Administration	84
SMS Server Supports SMPP v3.4 Operations	86
Digital Administration	87
Managing Sensitive Data	90
Masking Sensitive Data	101
General Recommendations	103
Database Performance	104
Unicode Character Support	105
Classification Server	106
Security	107

eServices Administrator's Guide

This guide presents recommendations for monitoring and adjusting your eServices configuration, plus detailed explanations of procedures to use for selected special purposes.

Most information is classified according to the software component that it applies to; there is also a section containing general information:

- [Interaction Server](#)
- [E-mail Server](#)
- [Universal Contact Server \(UCS\)](#)
- [Chat Server](#)
- [SMS Server](#)
- [Knowledge Manager](#)
- [Privacy Manager](#)
- [General recommendations](#)

See also a description of [limitations on multi-tenancy](#).

Important

For the latest version of Interaction Server Deployment Guide, see [here](#).

For the latest version of Interaction Server Administration Guide, see [here](#).

Interaction Server Administration

This section provides information for administrators regarding Interaction Server

In addition to the information on this page, there is also information on:

- [Limitations](#) to observe concerning Interaction Server.
- [Improving the performance](#) of the Interaction Server database.
- [Converting](#) attached data to and from BLOB format.
- Deploying and using [Event Logger](#), which stores reporting event messages in a database.

Be aware of the following:

- Use CC Pulse to monitor interaction queues (in interaction workflows) for signs of problems with routing strategies. If the number of interactions in a queue increases abnormally, it may be a sign that the strategy that processes interactions from that queue is not loaded in Universal Routing Server.
- Depending on the amount of configuration objects and the volume of the interactions stored in the Interaction Server database, it might take considerable time for Interaction Server to start up and shut down.
- Interaction Server has [two possible Application types](#) in the Configuration Layer. Interaction Server is the normal type; the T-Server type is also available for backward compatibility. Be aware that an Interaction Server 7.6 or later of type T-Server, upon startup, will make two attempts to connect to Configuration Server. The first attempt will generate trace-level alarms (about a missing application of type: Interaction Server) that you should ignore. The second attempt will succeed.
- If you want to use the Dynamic Workflow Management functionality, be sure to run Interaction Server with a user that has write access to the Configuration Server database for all of the tenants associated with this Interaction Server (that is, the user specified on the Security tab of the Interaction Server Application object).
In this situation Interaction Server does not support Configuration Server Proxy, which has only read access to the Configuration Server database.

Important

For the latest version of Interaction Server Deployment Guide, see [here](#).

For the latest version of Interaction Server Administration Guide, see [here](#).

Interaction Server Clusters

Starting with release 8.5.106.x of Interaction Server and 8.5.107.x of Interaction Server Proxy, you can [configure multiple Interaction Servers into a cluster](#) that works with a single instance of Interaction Server Proxy.

KPI Counters

Starting with the 8.5.102.02 release, Interaction Server includes KPI (Key Performance Indicator) counters that monitor:

- The number of requests of different types received from clients.
- The number of Interaction Server protocol errors, counted per error type, sent to clients.
- The number of ESP (External Service Protocol) errors, counted per original request type, sent to clients.

Interaction Server clients can access these counters using EventPing.

By default, Interaction Server does not calculate these counters. To enable the counters, set the following options in the **[settings]** section to true:

- For requests received, `collect-request-counters`
- For Interaction Server protocol errors, `collect-error-counters`
- For ESP errors, `collect-esp-error-counters`

Interaction Server Limitations

- Interaction Server does not support the following requests:
 - RequestQueryServer
 - RequestQueryLocation
 - RequestDeletePair (when URS sends this request after RequestRouteCall)
- It is not desirable to run Interaction Server in an environment in which servers and clients differ as to the codepages used (by operating systems or databases). In such an environment, characters of non-Latin alphabets may appear as the symbol ? (question mark) in log files and in applications with a user interface, such as Agent Desktop. The functionality of other features of the solution may also be restricted or compromised.
- Making an on-the-fly change to the host or port specification (on the `Server Info` tab) of a backup Interaction Server will cause it to exit.
- When Interaction Server sends a database request right before disconnecting from DB Server, and the request executes after disconnecting, Interaction Server fails to generate events to clients for submitted interactions.
- Starting in release 8.1.3, the scripts supplied with Interaction Server for Oracle databases create the `flexible_properties` field with the type BLOB. To support this feature, you must use DB server 8.1.1 and above with Oracle client 10.2 and above.
- Take [these precautions](#) when configuring Interaction Server for high availability (HA).
- The interaction state timeouts set by the following options are not exact and can be delayed for up to 10 seconds due to the way they are implemented.
 - `delivering-timeout`
 - `handling-timeout`
 - `routing-timeout`
- You cannot use commas (,) and semicolons (;) in interaction queue names.

Interaction Ordering in Clusters

For the ordering of interactions to work correctly in **RequestGetWorkbinContent**, **RequestFindInteractions** and **RequestTakeSnapshot** in cluster environments, the following conditions must be met:

- All Interaction Server nodes in the cluster must use the same type of database.
- The database encoding and collation must be the same for all the databases used by Interaction Server.
- The encoding used by all the cluster nodes must be the same. Genesys recommends using UTF-8 for Interaction Server. (Interaction Server Proxy neither requires nor allows setting UTF-8 specifically).
- The ordering may contain only predefined interaction properties (a list is provided below) and defined interaction custom properties. The field names may also be used interchangeably with the

corresponding property names.

In addition, these conditions are important because the Business Process definition (view conditions and orders) may be specific to the particular database type, and all nodes within the cluster use the same business process.

Below is a list of the field names and corresponding predefined interaction property names that can be used in the attributes that specify interaction order:

Field name	Property name
abandoned_at	AbandonedAt
assigned_at	AssignedAt
assigned_to	AssignedTo
completed_at	CompletedAt
delivered_at	DeliveredAt
external_id	ExternalId
held_at	HeldAt
id	InteractionId
is_locked	IsLocked
is_online	IsOnline
media_type	MediaType
moved_to_queue_at	MovedToQueueAt
parent_id	ParentId
place_in_queue_seq	PlaceInQueueSeq
placed_in_queue_at	PlacedInQueueAt
priority	Priority
queue	Queue
received_at	ReceivedAt
scheduled_at	ScheduledAt
service_objective	ServiceObjective
state	InteractionState
submit_seq	SubmitSeq
submitted_at	SubmittedAt
submitted_by	SubmittedBy
subtype	InteractionSubtype
tenant_id	TenantId
type	InteractionType
workbin	Workbin

Improving the Performance of the Interaction Server Database

To optimize the performance of Interaction Server, try the following steps:

1. Design or redesign the Business Process for greater efficiency; for example, by minimizing the number of processing steps. This provides for the most performance gain for custom Business Processes.
2. Analyze and optimize the SELECT statements generated by Interaction Server. Analyze the execution plans for the generated SELECT statements and create appropriate indexes. This is especially important if you have added an custom Business Processes: the standard indexes provided with the default schema do not take account of any custom database fields, specific conditions configured, and other items added by custom Business Processes. This step might provide all the performance gain that you need.
3. Perform a general tuneup on the database.
4. Partition the database. The subtopics in this section deal with this.

General Remarks on Partitioning

A partition is a division of a logical database or its constituent elements into independent parts. Database partitioning may be done for reasons of performance, manageability, or availability. This section concentrates on partitioning to improve performance.

By splitting a large table into several smaller tables, queries that need to access only a fraction of the data can run faster because there is less data to scan. Maintenance tasks, such as rebuilding indexes or backing up a table, can also run more quickly. Placing logical parts on physically separate hardware provides a major performance boost since all this hardware can perform operations in parallel.

Interaction Server performs large numbers of queries, updates, inserts, and deletes on its database. While it is relatively easy to achieve optimal performance with updates, inserts, and deletes, queries (SELECTs) are different.

The Interaction Server database consists of a single major table that stores all the interaction data. Every interaction in the system is always assigned to some interaction queue, represented by value of the field queue in the Interaction Server table. Business processes may employ dozens or even hundreds of queues.

Queues can vary greatly in the way they are used: some hold many interactions which are rarely processed at all (for example, an archive queue), others hold a small number of interactions with a high processing rate (for example, a queue for interactions that need some preliminary processing).

If these two types of queue are separated into different partitions, then the slower selection rate of the first type will not interfere with the high-speed selections of the second type. So the queue field is a natural choice to partition the data on. The remainder of this section describes partitioning by queue.

Planning

Decide for which queues it makes sense to separate data into logical partitions. Start by surveying the queues in your Business Processes and separate them out into three types:

1. Queues that contain high numbers of interactions; for example, post processing backlog or archive queues.
2. Queues that should not contain lots of interactions because all interactions in these queues should be processed immediately. A good example is the first queue in a Business Process which is meant for some preliminary processing (such as performing classification, calculating and attaching some user data, or sending an acknowledgment).
3. Queues that feed strategies that wait for resources (agents) to become available; usually there is a single such distribution queue in a Business Process.

Here is the rationale for separating data into at least three partitions that correspond to these three types of queues:

1. Separating Type 1 queues, those with many "inactive" interactions, ensures that these interactions are not even considered when SELECT statements are executed to pull interactions from Type 2 ("active") queues. Even if there are complex conditions for some views in your Business Process, there is much less data to scan because the majority of the interactions in an archive or post processing backlog are not touched by these scans.
2. Separating Type 2 queues is logical because most of the time these queues should be completely empty. Selecting new interactions out of these queues is trivial since there are not many interactions to select from.
3. Type 3 is the most demanding. While the rate of processing can be high, if there are many agents and handling time is relatively low, interactions may still accumulate in these queues when the peak inbound rate is higher than the processing rate. This means that SELECT statements are executed frequently against many records. If there are multiple queues of this type, it may be beneficial to assign them to separate partitions.

Hardware Planning

While purely logical separation of data may be of some benefit, placing the partitions on separate hard drives provides the best performance gain. In planning which drives in your system to dedicate to Interaction Server database partitions, it is advisable set aside one drive for the operating system and one for database log files, and place the Interaction Server database partition on other drives.

The rest of this section presents an example of partitioning using Microsoft SQL.

Creating the Database

To create a database with several file groups that will hold data for different partitions, use an SQL statement similar to the following:

```
CREATE DATABASE [itx_partitioned] ON PRIMARY
(NAME = N'itx802partitioned', FILENAME =
N'D:\MSSQL\DATA\itx802partitioned.ndf',
SIZE = 44828672KB, MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB),
FILEGROUP [P1]
(NAME = N'itx802partitioned1', FILENAME =
N'E:\MSSQL\DATA\itx802partitioned1.ndf',
SIZE = 2048KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB),
FILEGROUP [P2]
(NAME = N'itx802partitioned2', FILENAME =
N'F:\MSSQL\DATA\itx802partitioned2.ndf',
SIZE = 2048KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB),
FILEGROUP [P3]
(NAME = N'itx802partitioned3', FILENAME =
N'G:\MSSQL\DATA\itx802partitioned3.ndf',
SIZE = 2048KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
LOG ON
(NAME = N'itx802partitioned_log', FILENAME =
N'H:\MSSQL\DATA\itx802partitioned_log.ldf',
SIZE = 5095872KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO
```

Or you can create the database and file groups using Microsoft SQL Management Studio.

You can create as many file groups as your resources allow.

Carrying Out the Partitioning

Partition Function

The partition function calculates the logical partition number for any specific record based on the record's field value. We only need to consider the value of the 'queue' field since we want to partition data according to queues.

The following is an example of the partition function:

```
CREATE PARTITION FUNCTION [QNamePFN](varchar(64)) AS
RANGE RIGHT FOR VALUES (N'Archive', N'Distribution', N'Inbound')
GO
```

Note that an SQL server partition function is always a range function. The values for the range function must be sorted in ascending order so that you can clearly see which range any particular value falls into.

In the example above, all data that comes earlier in the alphabet than Archive is placed in the first partition. All data whose alphabetical order is the same or later than Archive but earlier than Distribution is placed in the second partition. All data whose alphabetical order is the same or later than Distribution but earlier than Inbound is placed in the third partition. All other data is placed in the fourth partition.

Note that it is the partition scheme that assigns a specific partition to the range; you can actually assign different ranges to the same partition.

Also, this partition function applies to all queues in the Business Process. For example, if there is a queue Begin it falls into the second range and will be assigned to the same partition as the Archive queue. But if Begin is not a Type 1 queue this result may be less than ideal. One way to ensure that every queue is assigned to its intended partition is to list all the queues in the Business Process in alphabetical order in the partitioning function, and then specify the appropriate partition for each queue. If a new queue is added to the Business Process, you can alter the partition function and partition scheme to take account of this new queue.

Partitioning Scheme

The partitioning scheme uses the partitioning function to define which records (with a particular value of the partitioning function) go to which partition.

```
CREATE PARTITION SCHEME [QNamePScheme]
AS PARTITION [QNamePFN] TO ([PRIMARY], [P1], [P2], [P3])
GO
```

Since our partitioning function is based solely on the value of the 'queue' field, our partitioning scheme tells the database which queue goes to which partition.

Partitioning the Table

To partition the table, simply specify the partitioning scheme for the table:

```
create table interactions
(
id varchar(16) not null,
...
) on QNamePScheme(queue)
Go
```

Note that we explicitly specify that the queue field value should be given to the partitioning scheme and subsequently to the partitioning function to decide which partition the record should go to.

Verification

The following SQL statement is an easy way to monitor how many records are stored in each partition for a given partitioned table (the interactions table in this example):

```
SELECT
p.partition_number, fg.name, p.rows
FROM
sys.partitions p
INNER JOIN sys.allocation_units au
ON au.container_id = p.hobt_id
INNER JOIN sys.filegroups fg
ON fg.data_space_id = au.data_space_id
WHERE
p.object_id = OBJECT_ID('interactions')
```

This produces results similar to those shown in the following table.

Partition_number	Name	Rows
1	PRIMARY	1
2	P1	1
3	P2	1
4	P3	1

The table shows that each partition contains a single record. If you insert a new record and execute the above statement again, it will show which partition the new record has been placed in, verifying your partition function and scheme.

Important

To compare performance of the partitioned database with an unpartitioned database, you will need to artificially create a certain distribution of interactions between partitions (different queues) and see how fast the same SELECTs are being executed. When interactions change queues, the records are physically relocated into different partitions (according to the partition scheme).

Converting to and From BLOB

Interaction Server ordinarily stores attached data in the `flexible_properties` field as a BLOB (binary large object).

Converting from BLOB

You can convert attached data to a custom field by running Interaction Server in a special utility mode, in which Interaction Server uses the key-value format of this attached data to convert all such fields to custom fields.

To run Interaction Server in utility mode, launch it from a command line with the following option:

```
-convert-fields [command_or_parameters]
```

where the optional `command_or_parameters` is one of the following:

- `reset`—Ensures that the next run in utility mode will start processing from the beginning, rather than picking up where it left off.
- `bulk-size=N`—Determines the number of records that are processed before committing the transaction. The default value is 100, valid values are any integer in the range 1-1000.

Here is an example command line:

```
interaction_server -host genesys_host -port 9876 -app IxnSrv05 -convert-fields reset
```

You can also have Interaction Server convert an existing database field into a BLOB, stored in the `flexible_properties` field. To do so, use the following procedure.

Converting a field to a BLOB

1. Open the corresponding Business Attribute Value in Configuration Manager.
2. In the **translation** section, add an option called `to-delete` and give it the value `yes`.
3. Run Interaction Server in utility mode, as described above. Interaction Server, in utility mode, moves the content of all such fields into the `flexible_properties` field and leaves the custom field with an empty value.

Important

When Interaction Server runs in utility mode all of its other features are disabled: it cannot process interactions or open ports for clients.

Event Logger

In release 7.6.1 and later, Interaction Server includes Event Logger, a mechanism for storing reporting event messages in a database or a [message queue](#). You can configure it to store all reporting events or a selected subset. You can also create multiple instances of it.

Interaction Server generates, to registered reporting engines, messages that provide a detailed picture of the processing of each interaction. It [classifies](#) these messages, in two ways. The attributes of these messages include much information about the interaction itself, such as its type, time received, associated agents, queues and workbins it was placed in, and so on. For a reference listing of these events and their attributes, see the “Reporting Messages” section of the “Other Protocol Events Used by Interaction Server” chapter of the [Genesys Events and Models Reference Manual](#).

All [configuration](#) for the logger functionality is done in the Database Access Point (DAP) associated with the logger database.

There are ways to [manage](#) the flow of data produced by Event Logger.

Deploying Event Logger

1. Create a database to store the reporting data.
2. Locate the correct setup script for your RDBMS and run it on the database you created in Step 1. This script is called `eldb_<database_name>.sql`, where `<database_name>` is either *db2*, *postgre*, *mssql*, or *oracle* (for example, `eldb_mssql.sql`). To locate the script, go to the Script subdirectory of the installation directory of your Interaction Server, then open the subdirectory named after your RDBMS; for example, `\InteractionServer_801\Script\Oracle`.
3. Create a Database Access Point (DAP), filling in the usual mandatory settings on the General and DB Info tabs.
4. On the DAP's Options tab, create a section called `logger-settings`. This is the only mandatory section; its existence tells Interaction Server to use this DAP for storing reporting events.
5. In the `logger-settings` section, add at least one **option** (the section must contain at least one option in order to be valid).
6. Optionally add any of the following section types:
 - `event-filtering`—Contains options filtering out certain classes of event messages
 - `custom-events`—Specifies a custom mapping of the `CustomEventId` attribute value of `EventCustomReporting` (the option name) to the Event Logger table to store them in (the option values)
 - `Custom data sections`—Five sections that enable you to map the name of any event onto a **custom field** in the Logger database.
7. On Interaction Server's Connections tab, add a connection to the DAP. For multiple instances of the Event Logger, run the creation script multiple times, creating multiple databases. Also create a DAP for each database.

Managing Event Logger Data

For the `rpt_interaction`, `rpt_agent`, and `rpt_esp` tables, Genesys supplies a set of scripts that deletes events as soon as processing of the interaction stops, the agent logs out, or the external service responds, respectively. For custom reporting events that are stored in the `rpt_custom` table, the event-driven trigger `trg_del_cust_delay` purges them from the `rpt_custom` table, with a configurable delay (the default is 10 minutes).

If you want to preserve this data, you can disable the triggers `trg_delete_stopped`, `trg_delete_resp`, `trg_del_cust_delay`, and `trg_delete_logout` after you run the setup script. For Oracle, additionally, disable the triggers `trg_mark_cust_logged`, `trg_mark_responded`, `trg_mark_ended_session` and `trg_mark_stopped_ixn`.

You can reenble the triggers any time and resume removing records from the database automatically.

Of course event messages increase rapidly in number as interactions are processed, so you will want to take measures to periodically delete data from the database or move it elsewhere.

Also note that after creating or removing custom fields in a database, some triggers become invalid. If this happens, you must recompile them to be sure they work properly.

Classification of Events in Event Logger

The logger functionality classifies reporting events in two ways:

- By activity type—that is, whether the activity refers to an interaction, an agent, an **ESP server**, or is of a custom type. The database contains tables for each type: interaction activity is stored in `rpt_interaction`, agent activity is stored in `rpt_agent`, and ESP server activity is stored in `rpt_esp`. Custom activity can be stored in `rpt_interaction`, `rpt_agent`, or `rpt_custom`, depending on the configuration in the custom-events section of the Event Logger DAP.
- By endpoint type—that is, whether that interaction is being transmitted to a queue, strategy, agent, or ESP service. You can **filter out events** according to endpoint type. A few events do not have an endpoint type; you cannot filter these events.

The following table lists the events and their classifications.

Event	Activity	Endpoint
EventPropertiesChanged	Interaction	-
EventPartyAdded	Interaction	Agent, Strategy
EventPartyRemoved	Interaction	Agent, Strategy
EventRevoked	Interaction	Agent
EventInteractionSubmitted	Interaction	-
EventProcessingStopped	Interaction	-
EventHeld	Interaction	-
EventResumed	Interaction	-
EventPlacedInQueue	Interaction	Queue
EventPlacedInWorkbin	Interaction	Queue
EventAgentInvited	Interaction	Agent
EventRejected	Interaction	Agent
EventTakenFromQueue	Interaction	Queue
EventTakenFromWorkbin	Interaction	Queue
EventAgentLogin	Agent	Agent State
EventAgentLogout	Agent	Agent State
EventDoNotDisturbOn	Agent	Agent State
EventDoNotDisturbOff	Agent	Agent State
EventMediaAdded	Agent	Agent State
EventMediaRemoved	Agent	Agent State
EventNotReadyForMedia	Agent	Agent State
EventReadyForMedia	Agent	Agent State
EventAgentStateReasonChanged	Agent	Agent State
EventMediaStateReasonChanged	Agent	Agent State

Event	Activity	Endpoint
EventExternalServiceRequested	ESP Server	ESP Server
EventExternalServiceResponded	ESP Server	ESP Server
EventCustomReporting	Interaction, Agent, or Custom	-

Event Logger Options

This section provides short descriptions of the DAP options that configure the Event Logger's behavior. See the [Event Logger DAP](#) section of the [eServices Options Reference](#) for full details.

logger-settings Section

`batch-size`—Defines the minimum number of records to store in internal memory before flushing to the database. Valid values are 1-5,000; the default is 500.

`max-queue-size`—Defines the maximum number of records that are kept in memory while waiting to be written to the database. If the number of records exceeds this maximum, the data are discarded from memory and are not written to the database. Valid values are 10,000-100,000; the default is 20,000.

`storing-timeout`—Defines a time interval, in milliseconds, between operations of writing to the database. Valid values are 500-60,000; the default is 1,000.

Important

`storing-timeout` and `batch-size` define limits that trigger writing to the database: writing takes place as soon as one or the other is reached.

`schema-name`—Specifies the name of the schema used to access the database.

custom-events Section

This section contains options that list custom events by their identifiers and specify which table (interaction, agent or custom) stores them.

event-filtering Section

This section contains eight options, six of which are named for one of the endpoint types that is referred to in the [classification of events](#):

`log-agent-state`
`log-agent-activity`

```
log-queue
log-strategy
log-esp-service
event-filter-by-id
```

With the value `false`, events associated with the named endpoint type are filtered out. For example, setting `log-queue` to a value of `false` prevents the events `EventPlacedInQueue`, `EventPlacedInWorkbin`, `EventTakenFromQueue`, and `EventTakenFromWorkbin` from being stored. The remaining two options in this section are:

- `log-userdata`—With the value `false`, data from custom fields is filtered out.
- `event-filter-by-id`—A list of comma-separated event identifiers. Only these events are stored in Event Logger. If this option is not present or contains no event identifiers, event filtering by identifier is not applied.

These event identifiers are listed in the [Platform SDK 8.1 API References for Java and .NET](#). For example, the identifier of `EventRejected` is 168.

Custom Data Sections

The five sections contain options specifying a list of events that are to be stored in custom fields of the event logger database. All five work identically, the differences being (a) the events from which the user data is taken and (b) the database table that stores them. These differences are shown in the following table.

Section	Source Event	Logger Database Table
<code>itx-custom-data</code>	UserData and EventContent attributes of interaction-related reporting events	<code>rpt_interaction</code>
<code>esp-custom-data</code>	UserData attribute of EventExternalServiceRequested and EventExternalServiceResponded	<code>rpt_esp</code>
<code>esp-service-data</code>	Envelope3rdServer attribute of EventExternalServiceRequested and EventExternalServiceResponded	<code>rpt_esp</code>
<code>agent-custom-data</code>	EventContent attribute of EventCustomReporting	<code>rpt_agent</code>
<code>custom-customdata</code>	EventContent attribute of EventCustomReporting	<code>rpt_custom</code>

For an explanation of the `Envelope3rdServer` attribute, see "EventExternalServiceRequested Attributes" and "Structure of Envelope3rdServer Attribute" in the "Reporting Messages" section of the "Other Protocol Events Used by Interaction Server" chapter of the [Genesys Events and Models Reference Manual](#).

To use these options, you must first add a field to the appropriate Event Logger database table. Its data type must be the same as that of the mapped user data key. In these sections, the options have

the following characteristics:

- The name is a user data key name (case-sensitive).
- The value is three semicolon-separated strings, which specify the following:
 1. The name of the field that you added to the database table. This value is required.
 2. The data type: string, integer, or timestamp. The default is string, with default length 64. If your data type is other than string, or if it is string and you want to specify a non-default length (next item), this value is required.
 3. Optionally, the length. The default for the string type is 64. There are no default values for integer and timestamp.

For example, if you have a data key called `CustomerSegment`, you can add a custom field to store this data as follows:

1. Add a field called `customer_segment` to the `rpt_interaction` table.
2. In the `itx-custom-data` section, create an option called `CustomerSegment`.
3. Give it this value: `customer_segment;string;64`.

Since `string` and `64` are the default values for type and length respectively, the value of this option could also be simply `customer_segment`.

Using a Message Queue with Event Logger

You can have Event Logger send events to a message queue, such as IBM MQ-Series, or Microsoft Message Queue (MSMQ). This provides a mechanism for reliable reporting events delivery to Interaction Server's reporting clients. Disconnection of the client does not lead to a loss of reporting events. Instead, events are stored in the message queue and delivered to the client (or otherwise read by the client) after it reconnects.

To use this functionality, you must create a DAP object that is specifically for the streaming of reporting events into MSMQ or MQ-Series. Both Interaction Server and the client connect to this DAP. This DAP must have the following section and options, which partly resemble the sections and options of the Event Logger DAP and are also documented in the [eServices 8.1 Reference Manual](#):

- `logger-settings` section
 - `delivery-protocol`. Possible values are:
 - `event-log`—The default, for using Event Logger database scripts
 - `mq-series`—For the MQ-Series message queue system
 - `msmq`—For the MSMQ message queue system
 - `jms`—For a [JMS](#) queue
 - `groovy`—For the [Groovy Event Logger](#)
 - `delivery-queue-name`—The name of the queue to send messages to.

Optionally, Event Logger DAP can use the following section and option:

- `event-filtering` section. Contains the single option `event-filter-by-id`, whose value is a list of comma-separated event identifiers. Only these events are sent to the message queue; events not listed are not sent. This option is analogous to the option of the same name used in the Event Logger DAP.

Important

The event identifiers used in `event-filter-by-id` are listed in the [Platform SDK 8.1 API References for Java and .NET](#).

Using the JMS Event Logger with Apache ActiveMQ

Important

You must have Apache ActiveMQ 5.14.5 or higher.

You can use the JMS Event Logger with Apache ActiveMQ to process reporting events using a JMS queue.

To enable the JMS Event Logger with Apache ActiveMQ, [edit the option **delivery-protocol**](#) and set the value to `jms`.

Configuring JMS Event Logger with Apache ActiveMQ

1. Configure Apache ActiveMQ, as described on the [Apache website](#).
2. Create a **jndi.properties** file with the following content:

```
java.naming.factory.initial = org.apache.activemq.jndi.ActiveMQInitialContextFactory
java.naming.provider.url = tcp://activemq_host:61616
connectionFactoryNames = ConnectionFactory
queue.InxEventLogQueue = InxEventLogQueue
queue.inbound = inx.inbound
queue.error = inx.error
queue.processed = inx.processed
queue.notification = inx.notification
```

Important

The value of **queue.InxEventLogQueue** refers to a queue name in your environment.

3. Pack the **jndi.properties** file in the root of **amq-jndi.jar**. You can use a ZIP archiving utility to pack the `jndi.properties` file.
4. In the Interaction Server Application object, on the **Applications Options** tab, add the following jar files to the `-Djava.class.path` option in the **jvm-options** section:
 1. **activemq-all-<version>.jar**
 2. **amq-jndi.jar**
 3. **log4j-api-<version>.jar** (optional, refer to the Note below)

The **activemq-all-<version>.jar** file is located in the ActiveMQ installation directory and not supplied

with Interaction Server.

Important

Since version 5.17.0 **activemq-all** also requires log4j-api library, refer to [Maven Repository: org.apache.activemq » activemq-all](#) for information about log4j-api version. The **log4j-api-<version>.jar** file is not supplied with Interaction Server and should be downloaded from public repositories (For example, [Maven Repository: org.apache.logging.log4j » log4j-api](#)).

5. Create the JMS Event Logger as a [Database Access Point \(DAP\)](#).
6. Configure JMS Event Logger for connecting to Apache ActiveMQ by adding the following options in the **logger-settings** section:
 - `delivery-protocol=jms`
 - `delivery-queue-name=InxEventLogQueue`
 - `jms-connection-factory-lookup-name=ConnectionFactory`
 - `jms-initial-context-factory=org.apache.activemq.jndi.ActiveMQInitialContextFactory`
 - `jms-provider-url=tcp://activemq_host:61616`
 - `reconnect-timeout=10`

Important

- Change the value of **InxEventLogQueue** to the queue value you set in the **jndi.properties** file.
- The value of **jms-provider-url** is the Apache ActiveMQ URL that corresponds to the `<transportConnectorname>` node from the **activemq.xml** file.

7. Optionally, you can repeat the steps in this section to set up multiple JMS Event Loggers. With each additional logger, you must increment the name of additional queues as `queue.InxEventLogQueue2 = InxEventLogQueue2` in the **jndi.properties** file.
8. Add created JMS Event Loggers to Interaction Server connections.

Using TLS with Apache ActiveMQ

1. Prepare the TLS certificates, as described in the [Genesys Security Deployment Guide](#).
2. Copy **cert.jks** and **truststore.jks** into the following folder: **<Apache ActiveMQ installation directory>/conf**.
3. Open the file **activemq.xml** in the folder **<Apache ActiveMQ installation directory>/conf** and add the following lines:

```
<transportConnectors>
```

```
...
<transportConnectorname="ssl"uri="ssl://0.0.0.0:61617?trace=true&needClientAuth=true"/>
...
</transportConnectors>
<sslContext>
  <sslContextkeyStore="file:${activemq.base}/conf/cert.jks"
  keyStorePassword="YourKeyStorePassword"
  trustStore="file:${activemq.base}/conf/truststore.jks"
  trustStorePassword="YourTrustStorePassword"/>
</sslContext>
```

Important

Change the values of **keyStorePassword** and **trustStorePassword** to acceptable passwords.

- Restart ActiveMQ.
- Update the following configuration options in the **logger-settings** section:
 - `jms-initial-context-factory=org.apache.activemq.jndi.ActiveMQSslInitialContextFactory`
 - `jms-provider-url=ssl://activemq_host:61617`
- Add the following configuration options in the **jms-additional-context-attributes** section:
 - `connection.ConnectionFactory.keyStore=cert.jks`
 - `connection.ConnectionFactory.keyStorePassword=keyStorePassword`
 - `connection.ConnectionFactory.keyStoreType=jks`
 - `connection.ConnectionFactory.trustStore=truststore.jks`
 - `connection.ConnectionFactory.trustStorePassword=trustStorePassword`
 - `connection.ConnectionFactory.trustStoreType=jks`

Important

Change the values of **connection.ConnectionFactory.keyStorePassword** and **ConnectionFactory.trustStorePassword** to the values you set in the **activemq.xml** file.

Using the Groovy Event Logger

You can use the Groovy Event Logger to process reporting events using custom Groovy scripts.

The Groovy Event Logger is configured, as any other event logger, using the Database Access Point application and has its own specific options.

To enable the Groovy Event Logger, **edit the option `delivery-protocol`** and set the value to groovy.

The Interaction Server IP (installation package) contains sample Groovy projects that implement the following types of event loggers:

- Elasticsearch Event Logger—Logs the reporting events to Elasticsearch.
- Kafka Event Logger—Logs reporting events to the Kafka message queue.
- File Event Logger—Simplified sample that logs the reporting events to files. Only use this project as a starting point or for integration with third-party systems.

The IP also contains a document that describes the sample projects, as well as an XML template file for each sample project.

Using the Groovy Event Logger with Apache Kafka

You can use the Groovy Event Logger to send reporting events to an Apache Kafka topic.

Prerequisites

The following prerequisites must be met in order to enable sending of the reporting events to Apache Kafka via Groovy Event Logger:

- Interaction Server 8.5.306+ must be installed.
- Existing Kafka cluster must be present. It should be running brokers of version “0.10.1.0” or newer.
- Minimum JRE 8 is required.
- OpenJDK 11 is recommended.
- Sending of the reporting events to Apache Kafka is supported by using Groovy Event Logger and Java Kafka client and requires Java in order to work. If Java is not installed or not properly configured, this functionality will not be available. All required jar files are provided in IP. Java should be correctly configured in Interaction Server options. See [Java configuration](#) in the Integrated Capture Point Guide.

Configure Interaction Server application object

Add the following jars from the Interaction Server installation folder to the `-Djava.class.path` option in the `jvm-options` section:

- `lib/ixn-java-aux.jar`
- `lib/groovy-all-2.4.15.jar`

Configure Kafka Groovy Event Logger application object

To configure Kafka Groovy Event Logger application object:

1. Import the application template from `GroovyEventLoggerTemplates/KafkaGroovyEventLogger.apd` from the Interaction Server installation folder.
2. Create a new application using the newly imported application template.
3. Fill in the usual mandatory settings on the General and DB Info tabs. Following mandatory settings are not used and can be filled with the arbitrary values like:

- DBMS Name = groovy
 - DBMS Type = DB2
 - Username = groovy
4. Verify value of `logger-settings/script-classpath` points to `lib/KafkaEventLogger` in the Interaction Server installation folder.
 5. Specify the following options in the new application:
 - `kafka-settings/servers` - comma-separated list of the Kafka broker(s) to connect to initially
 - `kafka-settings/topic` - topic to send reporting events to
 6. Specify the following option in the new application if you want to send messages in Genesys binary format rather than in JSON (default)
 - `kafka-settings/binary-output=true`
 7. On Interaction Server's **Connections** tab, add a connection to the DAP.

Advanced Java Kafka Client Options

Java Kafka client used for communication with the Kafka cluster has many configurable options. Options specified in the `producer-options` section are passed to the Java Kafka producer as is. Producer is used to send message to the Kafka cluster. The list of options can be found on the [official page](#). Any of these options can be configured in this section except `bootstrap.server`. Value of `kafka-settings/servers` is used for the latter option.

Using the Groovy Event Logger with Elasticsearch

You can use the Groovy Event Logger to send reporting events to Elasticsearch cluster.

Prerequisites

The following prerequisites must be met in order to send reporting events to Elasticsearch via Groovy Event Logger:

- Interaction Server 8.5.306+ must be installed.
- Elasticsearch cluster of version "7.4.2" or greater must be installed and running in good health condition. You can verify the cluster's health by accessing the URL `http://HOSTNAME:9200`.
- Minimum Java 8 is required.
- OpenJDK 11 is recommended.
- Java is a primary requirement for installing and running Elasticsearch. If Java is not installed or not properly configured, Elasticsearch cluster cannot be started.
- All other required jar files are provided in IP. Java should be correctly configured in the Interaction Server options. See [Java configuration](#) in the Integrated Capture Point Guide.

Configure Interaction Server application object

Add the following jars from the Interaction Server installation folder to the **-Djava.class.path** option in the **jvm-options** section:

- `lib/ixn-java-aux.jar`
- `lib/groovy-all-2.4.15.jar`

Configure Elasticsearch Groovy Event Logger application object

To configure Elasticsearch Groovy Event Logger application object:

1. Import the application template from **GroovyEventLoggerTemplates/ElasticsearchGroovyEventLogger.apd** from the Interaction Server installation folder.
2. Create a new application using the newly imported application template.
3. Fill in the usual mandatory settings on the **General** and **DB Info** tabs. Following mandatory settings are

not used and can be filled with the arbitrary values like:

- **DBMS Name** = groovy
 - **DBMS Type** = DB2
 - **Username** = groovy
4. Verify the value of `logger-settings/script-classpath` points to `lib/ElasticsearchEventLogger` in the Interaction Server installation folder.
 5. Specify the URL where your Elasticsearch cluster is running in the **elasticsearch-settings/cluster-uri** option in the new application. For example, `http://localhost:9200`.
 6. On the Interaction Server's **Connections** tab, add a connection to the Database Access Point (DAP).

Important

Binary data indexing is disabled in Elasticsearch event logger because Elasticsearch have Base64 encoded string representation and analyzing the Base64 string makes Elasticsearch create more searchable tokens (sometimes exceeding max number of tokens).

E-mail Server Administration

Warning

This content has been moved to the *E-Mail Server Administration Guide* as of October 29, 2018 and, as such, the E-mail Server content in this guide will no longer be maintained.

JavaMail Properties

Warning

This content has been moved to the *E-Mail Server Administration Guide* as of October 29, 2018 and, as such, the E-mail Server content in this guide will no longer be maintained.

Delivery Status Notification and Message Disposition Notification

Warning

This content has been moved to the *E-Mail Server Administration Guide* as of October 29, 2018 and, as such, the E-mail Server content in this guide will no longer be maintained.

Customizing the Format of External Resource E-mails

Warning

This content has been moved to the *E-Mail Server Administration Guide* as of October 29, 2018 and, as such, the E-mail Server content in this guide will no longer be maintained.

CSS for HTML Chat Transcripts

Warning

This content has been moved to the *E-Mail Server Administration Guide* as of October 29, 2018 and, as such, the E-mail Server content in this guide will no longer be maintained.

UCS Administration

This section provides information for administrators regarding Universal Contact Server (UCS). In addition to the topics on this page, there is also the following:

- Special instructions on **enabling a user who is not the schema owner** to run UCS with an Oracle database
- **Queries** that users of the UCS database must have permission to run
- **Updating interaction data** in a routing strategy
- **Limitations** to observe in operating UCS
- Enabling **full text searching** of the UCS database
- **Masking sensitive data**, such as credit card numbers, phone numbers, and Social Security numbers, in UCS logs.

Access to Configuration Server

Be sure to run UCS with a user that has write access to the Configuration Server database for all the tenants associated with this UCS (that is, the user specified on the **Security** tab of the UCS Application object).

This means that UCS does not support Configuration Server Proxy version 8.0.2 and earlier, which has only read access to the Configuration Server database. UCS does support Configuration Server Proxy version 8.0.3 and later.

Client Connection Timeout

To avoid inconsistency, every client of UCS should have the timeout of its connection to UCS set to a higher value than the timeout of UCS's connection to its Database Access Point (DAP).

This allows UCS to consistently either perform long queries or abort them, in accord with the clients' requirements.

Contact Identification and Creation

When a new interaction enters the system, UCS performs the following tasks:

1. **Contact identification**—UCS checks whether this interaction is coming from a known contact: more precisely, whether the contact data included in the new interaction matches an existing contact in the

UCS database. UCS does this in response to a request from a media server, the Identify Contact Routing strategy object, or the Agent Interaction SDK CreateInteraction method.

2. **Contact creation**—If the contact does not exist in the database, UCS creates a new record to represent it.

Character Sets

Oracle

The character set WE8ISO8859P1 does not have any representation of characters in the range 128-159. Because of this, with an Oracle database, attempting to save characters in this range in a column of type NCHAR or NVARCHAR results in corrupted data. Genesys recommends that for non-Unicode deployment you set the Oracle NLS_CHARACTERSET parameter to WE8MSWIN1252 instead of WE8ISO8859P1. WE8MSWIN1252 is a superset of WE8ISO8859P1, so there will be no data loss for most European languages (but data in non-Latin writing systems will still be corrupted).

For support of non-Latin charsets, use the following parameter settings in Oracle:

```
NLS_CHARACTERSET AL32UTF8
```

```
NLS_NCHAR_CHARACTERSET AL16UTF16
```

DB2 and PostgreSQL

DB2 and PostgreSQL must use the UTF-8 codeset for the UCS database.

Important

UCS support of PostgreSQL is limited and does not support the following:

- Multiple-site deployment
- Multi-tenant deployment
- Clusters
- High availability
- Load balancing

TLS and IPv6

The following two items also apply to E-mail Server.

TLS Connection as Windows Service

When UCS has Transport Layer Security (TLS) configured, either as a server on its ESP port, or as a client in its connection to Message Server, there are two ways to enable it as a Windows Service:

Log on As a Local Host User

1. Select the Windows service related to UCS.
2. Select the Log On tab. The default setting is Log on as local system account.
3. Select Log on as this account and provide the login/password of a local host user.

Import a Certificate to the Local System Account

Do one of the following:

- Run `psexec.exe -i -s mmc.exe`, then import a certificate for a user that is the local system account.
- Run `psexec.exe -i -s certutil -f -user -p [password] -importpfx [path to the certificate]`

Notes on `psexec.exe`:

- With the flag `-s`, `psexec.exe` executes the specified program under the system account.
- `psexec` is part of PStools, which can be downloaded from <http://technet.microsoft.com/en-US/sysinternals>

Mixing IPv6 and IPv4

UCS and E-mail Server do not support the command-line option `transport-ip-version`. If either of these is connected to Configuration Server using IPv6 but connected to other components using IPv4, you must add the following argument to `(ContactServerDriver|JavaEmailServerDriver).ini`:

```
-Djava.net.preferIPv6Addresses=true
```

Database Performance

OLTP

For best performance, Genesys strongly recommends that you set up the UCS database as OLTP (online transaction processing).

Tuning for Attachments

UCS uses the `Content` field of the `Document` table to store attachments; also, the `Content` field of the `ixnContent` table stores raw e-mails, including attachment data. If you plan to store large attachments (bigger than 5 MB), you should tune the database according to the recommendations of

your database vendor.

For example, increasing the block size of database files for these fields can greatly enhance performance in access and storing of large attachments, at the cost of a slight loss of performance with smaller ones. Also, some databases offer the ability to partition data according to specified criteria. Both tables have a `theSize` column that you can use to do such partitioning. This could enable you to store small attachments in a specific file and large ones in another, for example.

Refer to the tuning guides of your database vendor for more information.

User Other than Schema Owner

To enable a user who is not the schema owner to run UCS with an Oracle database:

1. Open the script `ucs_oracle_create_additional_user.sql`, which is located in the `sql_scripts` directory of UCS's starting directory.

2. Locate the following lines:

```
ucs_user := 'UCS_RUNTIME';  
ucs_db_creator := 'UCS_OWNER';
```

3. Replace `UCS_RUNTIME` with the name of the non-owner that you want to be able to run the database.

4. Replace `UCS_OWNER` with the name of the Oracle user that created (is the owner of) the schema.

5. Run the script from an Oracle account that has `SYSDBA` privileges. This creates the user identified in Step 2 and creates synonyms of all objects so that they are accessible to the newly created user.

6. Adjust the DAPs that UCS uses (or create new DAPs if you want to retain the existing ones), as follows:

- On the **DB Info** tab, set the **User Name** equal to the user identified in Step 2, and set the password equal to the user name. This is how the script creates the user and password. If you want a different password you must modify it in Oracle.
- On the **Options** tab, **settings** section, create the **db-schema-name** option. For its value, enter an upper-case version of the name of the Oracle user that created the UCS database schema (the user identified in Step 3).
- Also in the **settings** section, add an option called **includeSynonyms** and set it to `true`.

- After completing these steps on the main database, repeat them for the archive database.

Important

When using UCS with a limited DB user, UCS is not able to check for the existence of table indexes. The log will display a message warning that indexes do not exist, but you can ignore this warning. Using a limited user does not prevent DB from using indexes.

Required Queries

Users of the UCS DB must have permission to run certain queries on the database, for the following purposes:

- List user's tables—required for launching UCS
- Read NLS (national language support) configuration—required for normal operation of UCS
- List user's indexes—required for normal operation of UCS
- Read the configured maximum number of cursors—required for normal operation of UCS, on Oracle only

The following examples use `CONTACTSERV_USER` and `CONTACTSERVARC_USER` as the names of users of the main and archive databases respectively:

Oracle

To list user's tables:

```
SELECT T.TABLE_NAME, T.COLUMN_NAME, DECODE (T.NULLABLE, 'N', 'NO', 'YES') AS IS_NULLABLE
FROM ALL_TAB_COLUMNS T WHERE UPPER(T.OWNER)='CONTACTSERV_USER' ORDER BY T.COLUMN_ID
```

To read NLS configuration:

- In 8.1.0:

```
SELECT * FROM sys.props$ WHERE name LIKE 'NLS%CHARACTERSET%'
```

- In 8.1.1 and later:

```
SELECT * from SYS.NLS_DATABASE_PARAMETERS WHERE PARAMETER LIKE 'NLS%CHARACTERSET%
```

To list user's indexes:

```
SELECT I.INDEX_NAME, IND.COLUMN_NAME as COLUMN_NAME, I.UNIQUENESS as IS_UNIQUE FROM
USER_INDEXES I,
USER_IND_COLUMNS IND WHERE I.INDEX_NAME = IND.INDEX_NAME AND I.GENERATED='N' ORDER BY
INDEX_NAME
```

SQLServer

To list user's tables:

```
exec sp_tables @table_name = null, @table_type = ''TABLE''
```

To read NLS configuration:

```
SELECT DATABASEPROPERTYEX ('CONTACTSERV_USER', 'Collation')
```

To list user's indexes:

```
SELECT i.name INDEX_NAME, c.name AS COLUMN_NAME, CASE WHEN (i.status & 2)<>0 THEN 'true' ELSE  
'false'  
END AS IS_UNIQUE FROM sysindexes i INNER JOIN sysindexkeys k ON i.id=k.id AND i.indid=k.indid  
INNER JOIN syscolumns c  
ON c.id=i.id AND c.colid=k.colid WHERE INDEXPROPERTY (i.id , i.name , 'IsAutoStatistics' ) =  
0 ORDER BY index_name
```

DB2

To list user's tables:

```
SELECT TABNAME AS TABLE_NAME, COLNAME AS COLUMN_NAME, TYPENAME AS TYPE_NAME,  
LENGTH AS TYPE_LENGTH FROM syscat.columns WHERE UPPER(TABSCHEMA)='CONTACTSERV_USER' ORDER BY  
COLNO
```

To read NLS configuration:

```
SELECT TYPENAME, CODEPAGE FROM syscat.datatypes WHERE TYPENAME LIKE '%CHAR%' OR TYPENAME LIKE  
'%LOB%'
```

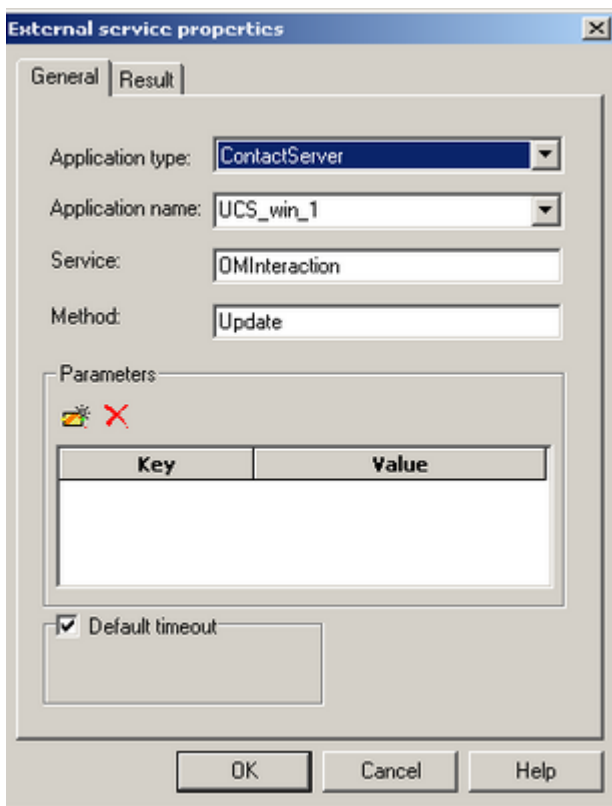
To List user's indexes:

```
SELECT INDNAME AS INDEX_NAME, COLNAMES AS COLUMN_NAME, UNIQUERULE AS IS_UNIQUE  
FROM syscat.indexes WHERE UPPER(TABSCHEMA)='CONTACTSERV_USER' ORDER BY INDNAME
```

Updating Interaction Data in a Routing Strategy

It is possible for a routing strategy to update certain interaction attributes. To ensure that such an update is also made in the UCS database, use the following procedure.

1. Log in to Interaction Routing Designer and open your strategy for editing.
2. In the strategy, create an External Service block to call the OMInteraction service, using the Update method. The figure below shows a sample configuration.



External Service Object

3. Save and reload the strategy.

UCS Limitations

Database

The Oracle SCAN (Single Client Access Name) feature is not supported.

Memory Allocation

By default Java allocates 512 MB of memory for UCS processes. This may not be enough when importing a large archive into Knowledge Manager, in which case an `OutOfMemoryError` exception may occur.

In such a case you can temporarily increase the amount of memory allocated for UCS processes by adjusting the Java option `-Xmx512m` to `-Xmx1000m`, as described in the following sections.

Windows

In the `ContactServerDriver.ini` file:

```
[JavaArgs]
-Xmx1000M
```

Unix

In the `contactServer.sh` file, change `-Xmx512M` to `-Xmx1000M` in the appropriate section, as in the following:

```
# Sun JVM</tt>
$JAVACMD -Xmx1000M -server -XX:+UseConcMarkSweepGC -XX:+UseParNewGC
-XX:+ExplicitGCInvokesConcurrent -Xbootclasspath/p:${JVM_ADDCHARSETS}
-Djava.rmi.dgc.leaseValue=60000 -Djava.library.path="${GMLLIB}"
-Dgenesys.cfllib="${LICENSE}" -Dtkv.multibytes="true"
-Djava.util.logging.config.file=${BASEDIR}/cv/jul.properties
-Dorg.restlet.engine.loggerFacadeClass=org.restlet.ext.slf4j.Slf4jLoggerFacade -classpath
"${COMP_CLASSPATH}" com.genesyslab.icc.contactserver.ContactServerEngine $*
else
# Ibm JVM
$JAVACMD -Xmx1000M -XX:+UseConcMarkSweepGC -XX:+UseParNewGC
-XX:+ExplicitGCInvokesConcurrent -Xbootclasspath/p:${JVM_ADDCHARSETS}
-Djava.rmi.dgc.leaseValue=60000 -Djava.library.path="${GMLLIB}"
-Dgenesys.cfllib="${LICENSE}" -Dtkv.multibytes="true"
-Djava.util.logging.config.file=${BASEDIR}/cv/jul.properties
-Dorg.restlet.engine.loggerFacadeClass=org.restlet.ext.slf4j.Slf4jLoggerFacade
-cclasspath "${COMP_CLASSPATH}" com.genesyslab.icc.contactserver.ContactServerEngine $*
```

There is a similar issue with [Knowledge Manager](#).

UCS Manager

UCS Manager (Universal Contact Server Manager) connects to UCS and provides a graphic interface that you can use to:

- Configure the options that handle maintenance of the UCS database.
- Correct certain problems that may exist with data integrity.
- Display statistics about the UCS database.

Important

Maintenance here refers to archiving and pruning.

[UCS Manager Help](#) describes how to perform all of these tasks.

Inactivity Timeout

If UCS Manager is inactive for a certain length of time, it first issues a warning, then requires you to log in again. The default length of this timeout is 15 minutes; the warning is issued two minutes before the time expires.

To adjust the length of the inactivity timeout,

1. In the Advanced View/Annex (Genesys Administrator) or the Annex tab (Configuration Manager), create a section called Security (if it does not already exist).
2. In the Security section, create an option called `inactivity-timeout` and give it the desired value. Valid values are any integer from 1 to 1440. Changes take effect upon restart.

Contact Identification

To perform contact identification, UCS takes the contact data included in the new interaction and tries to match it with its existing contact records based on certain attributes.

This page describes the default process of contact identification.

What UCS Checks

The default list of the attributes that UCS checks is `FirstName`, `LastName`, `Title`, `EmailAddress`, and `PhoneNumber`.

Important

This chapter refers to contact attributes by their system names. The system name is the one that is used in the UCS and Configuration Server databases, and the one that appears in the `Name` box of the attribute's `Properties` window in Configuration Manager. In Configuration Manager there is also a display name which usually differs slightly; for example, system name `PhoneNumber`, display name `Phone Number`.

Order of Checking

The attributes that UCS uses in contact identification have a ranking which tells UCS what priority to give them in searching. The default ranking is:

`EmailAddress`—0

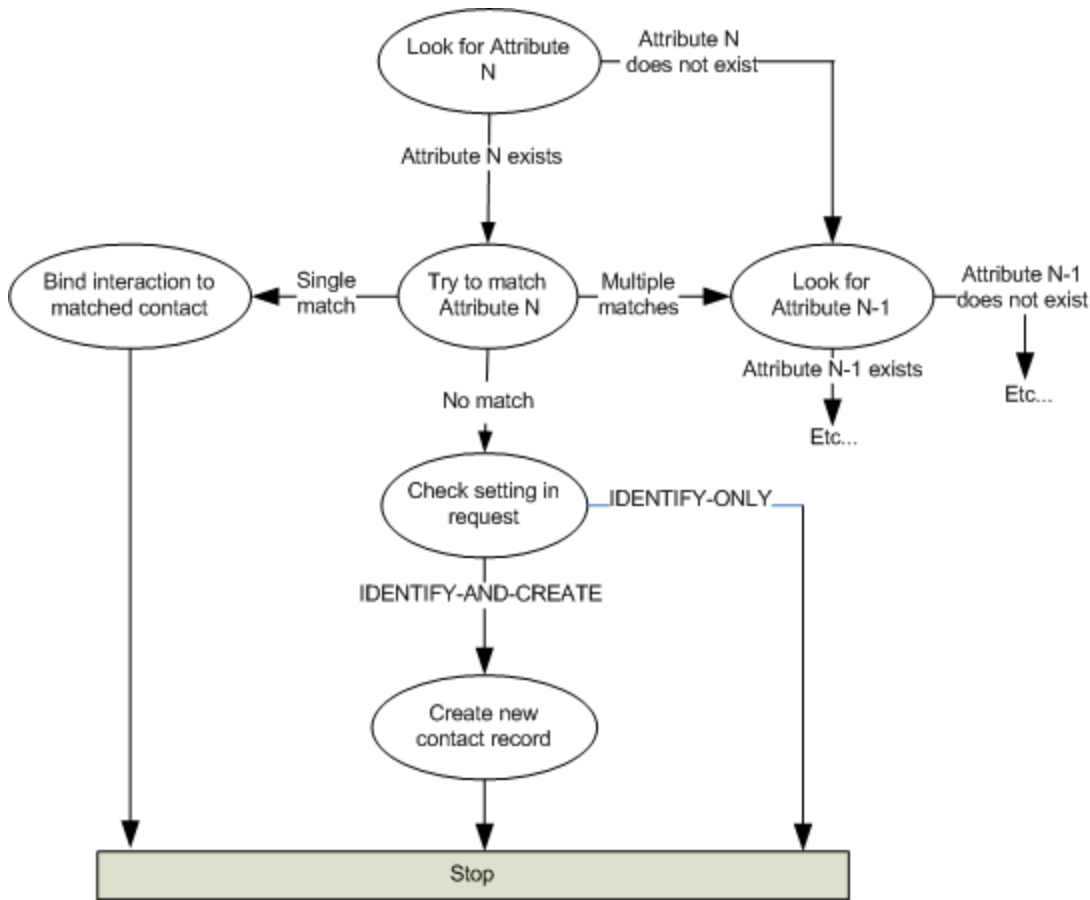
`PhoneNumber`—1

`FirstName`—2

`LastName`—2

`Title`—3

The general procedure that UCS follows in the default case can be diagrammed as in the figure below, where Attribute N is the highest-ranked attribute available in the interaction's contact data, Attribute N-1 is the next highest, and so on.



Default Process for Contact Identification

Example

An **example** of the default identification process is available.

Customizing

You can alter the default process of contact identification in both aspects:

- Whether an **attribute is used** in contact identification
- **What priority** an attribute has in matching

You can do this for all interactions globally or for interactions of a **specific media type**.

With these differing scopes of customization, the more specific scope takes precedence. UCS operates according to the following order:

1. If a media type has its own configuration for how a contact attribute is used in contact identification, that takes precedence over any other configuration.
2. Configuration of a contact attribute itself is more general and applies only to those media types that lack their own configuration for contact identification.
3. In any areas where items 1 and 2 do not apply, UCS follows the default behavior described on this page.

Contact Identification: Example

This page provides an example of the default process that UCS uses to **identify the contact** that is the source of a new incoming interaction.

1. A strategy requests identification of a contact, with only the following data provided:
 EmailAddress=johndora@doefamily.net
 FirstName=john
 LastName=doe
2. UCS takes EmailAddress, the highest priority attribute, and searches for EmailAddress = johndora@doefamily.net.
3. UCS finds two matching contacts, one with FirstName = john and the other with FirstName = dora.
4. UCS moves on to the next highest priority attribute. This would be PhoneNumber if it were specified, but it is not. So UCS takes the next-highest priority attribute, of which there are two, FirstName and LastName, both with a priority of 2.
5. UCS searches on EmailAddress = johndora@doefamily.net, and restricts the result to FirstName = john and LastName = doe
6. UCS finds a single contact matching these three criteria. It returns the ID of the Contact found, plus additional information if available.
 ContactId = 56464FGG519EA03
 EmailAddress = johndora@doefamily.net
 FirstName = john
 LastName = doe
 PhoneNumber = 555-654-6303

There are the following additional considerations:

- If the client's request conveys no specification of what to do in the event of no match, UCS creates a new contact record.
- When two attributes have the same rank, as do FirstName and LastName in the default ranking, UCS tries to match both of them.
 If the interaction has values for both attributes, UCS returns only contact records that match both. But if one of equally-ranked attributes has no value, UCS returns any contact records that match the attribute that does have a value (this is equivalent to saying that an attribute with no value matches everything).

If UCS goes through all of the attributes that it has been specified to check and finds no values for any of them, it returns an error message.

If UCS goes through all of the attributes that it has been specified to check and still finds two or more matching contacts, the next step depends on the entity that requested the contact identification:

- If UCS is identifying contacts in response to a request from a media server:
 - With E-mail Server, UCS takes the first contact on the list and associates it with the interaction.
 - With Chat Server, UCS simply passes the interaction on for processing with no associated contact. It reports neither the list of contacts found nor the fact that multiple contacts were found. The agent handling the interaction can select a contact for it using the Agent Desktop.

- If UCS is identifying contacts in response to an IRD `Identify Contact` object, it depends on whether the object's `Return Unique` checkbox is selected:
 - If `Return Unique` is not selected, UCS returns a list of the matching contacts.
 - If `Return Unique` is selected, UCS only reports that multiple contacts were found but does not return the list of matching contacts.

In either case, what happens next depends on the subsequent part of the strategy. The system may continue to process the interaction without any contact, until the interaction reaches the Agent Desktop, when the agent handling the interaction can select a contact for it.

Customizing

You can also **customize** this default behavior.

Specify Attributes To Check in Contact Identification

1. In Configuration Manager, open the `Properties` window for the desired contact attribute.
2. On the `Annex` tab of the `Properties` window (to set Configuration Manager to show the `Annex` tab, see [Configuration Manager Help](#)), create a `settings` section if it does not already exist.
3. In the `settings` section, create an option called `is-searchable`. Set its value to `TRUE` to make UCS use this attribute in contact identification. Set its value to `FALSE` to keep UCS from using this attribute in contact identification.

(The default value depends on the attribute. The five attributes used in UCS's [default behavior](#) have the default value `TRUE`. All other attributes have the default value `FALSE`).

Important

This procedure affects the value of the desired attribute's `IsSearchable` attribute in the UCS database. You must use only this procedure to do this. Never edit any attribute or value directly in the database.

There is also a way to control which attributes are searchable from the agent desktop, described in the "Making an Attached Attribute Sortable" section of the "Interaction Package" chapter of the [Genesys Events and Models Reference Manual](#)

Customize Attribute Priority

To customize the priority of the attributes that UCS checks, both of the following conditions must be true:

- The media type is specified in the interaction's user data.
- The interaction's media type is *not* e-mail, chat, or callback. In the typical case, this would be because the interactions are being submitted by a custom media server (built using the Genesys Media Interaction SDK Java or Open Media Interaction SDK Web Services).

To customize attribute priority,

1. In Configuration Manager, open the `Properties` window for the desired attribute and go to its `Annex` tab. Create a `settings` section if it does not already exist.
2. In the `settings` section, create an option called `search-order-level`. Give the option a numerical value to determine its priority in matching. Possible values are any integer in the range 0 (highest priority) to 127 (lowest priority). Two or more attributes can have the same priority.

Customize Contact Identification per Media Type

You can control how contacts are identified for interactions of a specific media type (of course, the interactions must have a valid media type specified in their user data).

1. In Configuration Manager, open the `Properties` window for the desired media type and go to its `Annex` tab.
2. Create a `contact-searchable-attributes` section.
3. In this section, create an option for each contact attribute that you want UCS to use in contact identification for this media type.

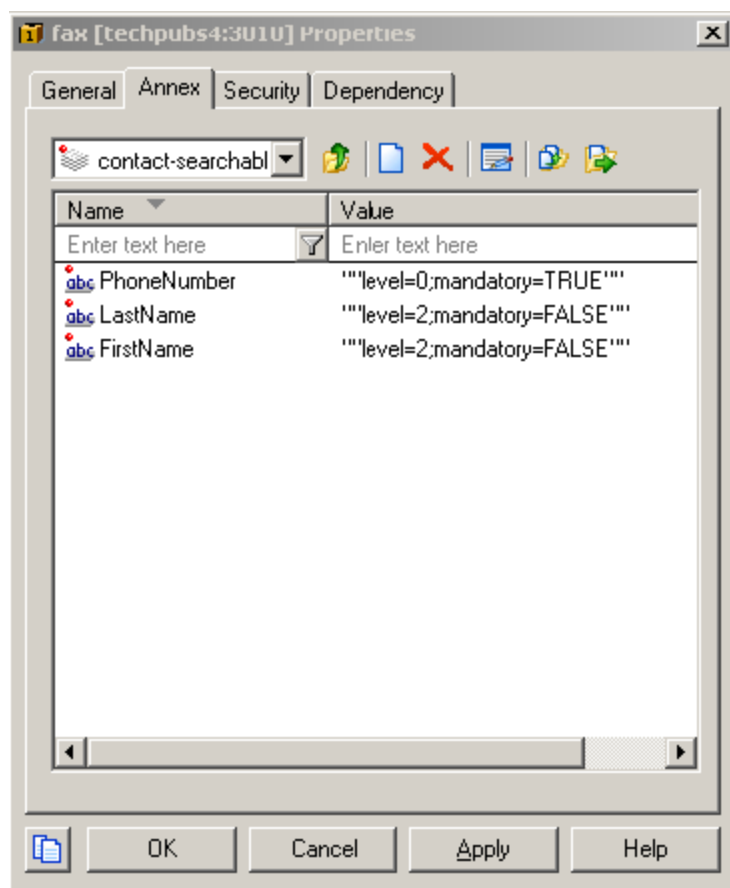
Important

Any such attribute must either be one on the [default list](#) or have the `is-searchable` option with a value of `TRUE`.

The options that you create must have the following characteristics:

- The name duplicates the attribute's system name (displayed in the `Name` box of the `Properties` window, not the `Display Name` box); for example, `PhoneNumber` rather than `Phone Number`.
- The value has the form `"level=<integer>;mandatory=<Boolean>"`, where
 - `<integer>` is an integer from 0-127 setting the attribute's priority.
 - If `<Boolean>` is `TRUE`, this attribute must be present (that is, it must have a value) for matching to occur.
 - If `<Boolean>` is `FALSE`, this attribute need not be present (that is, it may lack a value) for matching to occur.

The following figure shows an example.



Media Type Configured for Contact Identification

With the configuration in the preceding figure,

- UCS returns an error if no `PhoneNumber` value is present. It does this even if `FirstName` and `LastName` values are available, because `PhoneNumber` is the attribute with highest priority (0) and is defined as mandatory.
- If multiple matching records are found with a given `PhoneNumber` value, UCS uses `FirstName` and `LastName` together to discriminate because they have the same priority (2).
- If there is no `LastName` value, UCS does not use `FirstName` alone, because `LastName` is specified as mandatory in this level. Instead, the list of records matching `PhoneNumber` only is returned. This avoids identification based on `FirstName` only.

Contact Creation

If UCS cannot find a contact in its database that matches the contact associated with a new interaction, it creates one in its database.

Default Behavior

The default behavior is for UCS to simply create a new contact record. This behavior can be overridden for an interaction in a routing strategy using the IRD Identify Contact object (see [Universal Routing 8.1 Interaction Routing Designer Help](#)). For e-mail interactions only, you can also modify the default behavior by using one of the non-default settings of E-mail Server's contact-identification option (see the [eServices 8.1 Reference Manual](#)).

Turning contact creation on and off

You can customize the default behavior for each media type.

1. In Configuration Manager, go to Business Attributes > Media Type > Attribute Values and double click the media type that you want to adjust.
2. On the Annex tab of the resulting Properties window (to set Configuration Manager to show the Annex tab, see Configuration Manager Help), create a settings section if it does not already exist.
3. In the settings section, create an option called create-contact. Set its value to FALSE to block contact creation for this media type. (The default value is TRUE).

Setting minimum attributes for creation

You can also define a minimum set of contact attributes that must be present (must have values) for a contact to be created.

1. In Configuration Manager, open the Properties window for the desired media type and go to its Annex tab.
2. Create a contact-minimum-attributes-set section.
3. In this section, create an option for each contact attribute that you want to require for contact creation. The option's name must duplicate the attribute's database name (displayed in the Name box of the Properties window, not the Display Name box); for example, PhoneNumber rather than Phone Number. The option's value must be empty.

Searching the UCS Database

Overview

UCS supports full-text search of contacts, interactions, and standard responses using the Platform SDK (PSDK) Contact API.

To enable full text search, the stored data must be prepared, an operation known as indexing. Indexing converts, abbreviates, and sorts data to allow fast searching in large data sets.

There are various ways of preparing the data for the index; one way is to divide text data into words. Another way of preparing text data is *stemming*, or storing only the stem of a word rather than all its forms (for example, *run*, *running*, *runs*, and *ran* all stored as simply *run*). UCS uses *analyzers* to perform word division and stemming; you can **configure** which analyzer applies to which database field. In addition,

- **Examples** of the syntax used for searching are available.
- Many attributes are not directly searchable because of how they are stored; however you can **make such attributes searchable**.

Collocated and Distributed Deployment

This refers to the location of the indexes of the primary and backup UCS instances. The two types of deployment require different settings of the shared option (index section).

Collocated

In this type both primary and backup instances use the same storage path for the index. This is recommended in the following cases:

- For smaller deployments.
- Either both UCS instances are on the same machine or two instances on separate machines share the index files over a network.
- Read/write performance is not expected to be an issue.
- Network latency between UCS hosts and the shared file system is negligible

With collocated deployment, sharing must be enabled (`shared = true`) so that only the primary instance updates the index. It can also be enabled when sharing the index over a network.

Distributed

In this type each instance has its own set of index files. Note that the backup instance periodically

reads the database to replicate contact, interaction, and standard response updates to its local index. Here is a list of cases in which distributed deployment is recommended:

- For larger deployments.
- When the two UCS instances are on separate networks.
- For optimum index/search performance.
- In disaster recovery scenarios.

In this type of deployment, both UCS applications must have sharing disabled (`shared = false`).

More About Indexing

It is important to note that the UCS indexing service enables searching for text data, but does not guarantee its retrieval.

When an object is found it must be retrieved using standard PSDK methods. These methods are documented in the API References of the Platform SDK.

When designing an application, data coming from search results should be considered as cached data, like the preview that can be seen in web search engines. This data is optimized for search and cannot be used to update UCS database objects. For example, the `StructuredText` of an interaction is rendered from HTML to text. All formatting is lost if this data is re-inserted into the interaction.

Documents stored in the index will at least have the following retrievable fields:

- `id`—Unique identifier in the UCS database. This is what enables retrieval of the object from UCS.
- `IndexationDate`—The date this index was created. The format is UTC (`yyyy-MM-dd'T'HH:mm:ss.SSS'Z'`).
- `DocumentType`—Type of the document, selected from the following:
 - `EmailIn`
 - `EmailOut`
 - `PhoneCall`
 - `CallBack`
 - `CoBrowse`
 - `Chat`
 - `Interaction (Open Media)`
 - `Contact`
 - `StandardResponse`

Example:

```
id = 00001a4EM4TD007K
```

IndexationDate = 2013-10-02T13:54:54.318Z

Type = chat

In the index, even dates and numbers are text. This format enables alphabetic sorting, so that, for example, 21 December 2008 comes before 1 June 2009.

Default configuration enables the indexing of Contacts, Interactions and Standard Responses.

Sizing

Read about [sizing](#) for the database and its index.

Recommendations

- Distributed is recommended (index/shared=false).
- Store the index on a local dedicated SSD (Solid State Drive).
- If needed, run index rebuild while UCS is in backup mode, not primary mode. It is always better to rebuild off production hours.
- It is possible to copy a healthy index directory from one UCS to another while both instances are stopped.
- UCS Memory (XMX setting) may need to be increased if more than 3 months of data is kept in main database.

Sizing for the UCS Database and Index

The index that is produced by the UCS indexing service can be divided into two parts:

- The inverted index part that enables the full text search
- The data part that enables retrieval of the original data

If all of the original data is kept in the index, it can take as much space as the database itself; the default configuration does not compress it. The inverted index part is considerably smaller, but its size depends on the frequency of words. If the inverted index contains many unique values like IDs and timestamps it will be big. If it contains many common words such as those in the body of e-mails, it will be smaller.

The following table presents an example of sizing for a relatively large database.

Database Name	Database Size	Index Size	Duration
Standard responses	15 MB	6.5 MB	3 seconds
Contacts	5 GB	4.68 GB	2.5 hours
Interactions	20 GB	14 GB	2 hours

In this case the total size of the index files is about 19 GB.

During operations the index can grow to twice the size of the usable data. This is because index operations (such as internal reordering, purge of deleted documents, and concatenation) create new temporary files. To make these operations appear instantaneous, the system creates new files while it is still reading the old files. Then when it is finished creating the new files, it removes the old ones. Therefore, to be safe, free space on the disk hosting the indexes should be three times the size of the index.

Configuring Analyzers for UCS Search

Basic options controlling indexing and searching are described in the [eServices 8.1 Reference Manual](#). This page describes one further option that controls the use of analyzers.

Warning

Any change in configuration of indexing must be followed by a reindexing of the full content of all UCS indexed objects.

The analyzers that are supplied with UCS are

- **WhitespaceAnalyzer**—Splits the text into tokens separated by white space characters (specifically, SPACE_SEPARATOR, LINE_SEPARATOR, PARAGRAPH_SEPARATOR, HORIZONTAL TABULATION, LINE FEED, VERTICAL TABULATION, FORM FEED, CARRIAGE RETURN, FILE_SEPARATOR, GROUP_SEPARATOR, RECORD_SEPARATOR, or UNIT_SEPARATOR).
- **StandardAnalyzer**—Converts the text to lower case, splits the text into tokens separated by the white space character, and removes high-frequency English words (called *stop words*).
- **LowerCaseAnalyzer**—Converts the text to lower case and splits the text into tokens separated by the white space character.
- **SimpleAnalyzer**—Divides text at non-letters and converts to lower case. This works well for languages in which words are separated by spaces, such as most European languages, but is of little use for languages in which words are not separated by spaces, such as many Asian languages.
- **KeywordAnalyzer**—Treats the entire text as a single token. This is useful for data like zip codes, IDs, and some product names.

In the default case, UCS search uses the StandardAnalyzer for all fields in all tables in the database. To override the default analyzer, use the following option.

<table_name>-field-analyzer<any>

Optional: Yes

Default value: StandardAnalyzer

Valid values: See below

Changes take effect: After restart

Sets the analyzer used for any table or field. In the option name, <table_name> is one of the following tables in the UCS database:

- Callback
- Chat

- CoBrowse
- Contact
- ContactAttribute
- EmailIn
- EmailOut
- Interaction
- PhoneCall
- StandardResponse

<any> can be anything, including zero. Use it to differentiate among multiple field-analyzer options referring to the same table.

Values for this option have the general form

<field>=<analyzer>, <field>=<analyzer>, ...

where <field> is the name of a field in the table and <analyzer> is the name of a supported and installed analyzer. For example:

- Option name: interaction-field-analyzer
- Option value: Text=GermanAnalyzer,StructuredText=StandardAnalyzer

With this option name and value, when searching the Interaction table, the search operation applies GermanAnalyzer to the Text field and StandardAnalyzer to the StructuredText field.

You can achieve the same result by creating two options:

- Name:interaction-field-analyzer-01,value:Text=GermanAnalyzer
- Name:interaction-field-analyzer-02,value:StructureText=StandardAnalyzer

Supported Analyzers

General Analyzers

- WhitespaceAnalyzer
- LowerCaseAnalyzer
- SimpleAnalyzer
- KeywordAnalyzer

Language-specific Analyzers

These are the same as SimpleAnalyzer but also remove *stop words*: words that are so common that there is little to be gained in searching for them or listing their occurrences.

As an example, the stop words used by StandardAnalyzer, the language-specific analyzer for English, are *a an and are as at be but by for if in into is it no not of on or such that the their then there these they this to was will with*.

The language-specific analyzers installed with UCS are:

- BrazilianAnalyzer
- ChineseAnalyzer
- CJKAnalyzer (Chinese/Japanese/Korean; any language that uses Chinese characters/kanji/hanja)
- CzechAnalyzer
- DutchAnalyzer
- FrenchAnalyzer
- GermanAnalyzer
- GreekAnalyzer
- RussianAnalyzer
- StandardAnalyzer (English)
- ThaiAnalyzer
- SpanishAnalyzer
- ItalianAnalyzer

Note that `Interaction.AllAttributes` requires the usage of `KVListAnalyzer`, it is not possible to change it or customize it. The **AllAttribute** field is indexed as a text with following tokens: `._@-<>' "[]|=`

- It is not possible to do accurate wildcard matches on **AllAttributes**.
- It is difficult to do exact searches in the form `key=value`.

Search Syntax

Searching the UCS database uses the Lucene syntax, described at http://lucene.apache.org/core/3_6_0/queryparsersyntax.html.

Some samples follow.

```
FirstName:kristin*
```

This query searches for all records having a key FirstName whose value is any word starting with "kristin."

```
FirstName:Kim AND LastName:Brown AND EmailAddress:hotmail
```

This query searches on three attributes: FirstName containing "Kim," LastName containing "Brown," and EmailAddress containing "hotmail."

```
Text:complain*
```

This query searches for all records having an attribute Text which contains words starting with "complain."

Important

The first character of the search must not be * (asterisk) or ? (question mark).

Making an Attribute Searchable/Sortable from the Desktop

For both interactions and contacts, many attributes are not directly searchable from the desktop or cannot be used to sort results.

This section describes the way to make these attributes searchable from the desktop.

You do this by adding an option called `is-sortable` to the Annex section of the corresponding Business Attribute. Do not confuse this option with the `is-searchable` option. The following table compares the two.

Option	Effect	Used with
<code>is-sortable</code>	Makes attribute sortable from desktop, or returned in a list	Interaction attributes, contact attributes
<code>is-searchable</code>	Makes UCS use the attribute in identifying contacts (at runtime) or searchable from desktop	Interaction attributes, contact attributes

By default, all user data that are attached to an interaction are stored in the `AllAttributes` attribute of the Interaction entity. You can configure UCS to copy those attributes to one the 9 dedicated columns any attribute that is represented in the Configuration Server database by a Business Attributes object of type `Interaction Attributes`. This will allow interaction lists to be sorted by that attribute and/or to return the attribute as a column in the list. The Interaction entity includes attributes `StrAttribute1–StrAttribute10` and `IntAttribute1–IntAttribute5`, which exist to enable you to make attached attributes sortable. These attributes `StrAttribute1–StrAttribute10` and `IntAttribute1–IntAttribute5` may be referred to collectively as *replicant attributes*, as explained below.

Making an attached attribute searchable/sortable

1. In Configuration Manager, be sure that Properties windows show their Annex tabs. If they do not:
 - a. Go to the View menu and select Options.
 - b. In the resulting dialog box, select the Show Annex tab in object properties check box.
2. In the tenant for your UCS, go to Business Attributes > Interaction Attributes > Attribute Values.
3. Open the Properties window for the attribute that you want to make searchable (for example, Service Type).

4. On the Annex tab, create a section named settings if it does not already exist.
5. In this settings section, create an option named is-sortable and give it the value true.
6. If the attribute is of type string, you are finished. If it is of type integer, you must create an additional option, also in the Settings section, named type with the value integer.

Once you have configured an attached attribute as searchable/sortable, UCS takes its value as stored in AllAttributes and copies it as the value of one of the replicant attributes. To find out which replicant attribute copies a given attached attribute, look at the content of the IxnAttributeMetaData table. For example, if you have configured the ServiceType attribute to be searchable, you can find out which replicant attribute copies its value by using the following SQL request:

```
select MappingColumnName from IxnAttributeMetaData where TheName='ServiceType';
```

Please also note the following:

- This replication process only applies to interactions created or updated after you perform the configuration described in this section. The replication process is not applied to interactions retroactively. The replicant attribute in older records will remain empty.
- Replicant attributes are read-only from outside UCS. UCS is responsible for synchronization of their content whenever Interaction.AllAttributes is updated.
- The mapping between a searchable interaction attribute and a replicant attribute is based on the type (string or integer) of the business attribute declared in the Configuration Server database (string by default). UCS chooses from among the replicant attributes of the proper type that are not already associated with an attached attribute. It does this until no more replicant attributes are available.
- Once a replicant attribute has been used for a particular attribute, it is dedicated to that attribute: it cannot be used for another one. The only modification you can make is to configure a searchable attribute to be no longer searchable. The replicant attribute that copied this attribute's values will then retain those values for existing records and for any new records.

Chat Server Administration

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Sizing Guide, Setting Load Limits, and Health Monitoring

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Deploying a Chat Solution

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Deploying High-Availability Chat Server

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Configuring a secure connection to Cassandra

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Initialization Cassandra scripts for Chat Server

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Multilingual Processing in Chat Server

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Masking Sensitive Data

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Inactivity Monitoring

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Matching Contact Attributes

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

How to send ESP requests to Chat Session from Workflow

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Chat Server Reporting Statistics

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Chat Server API selected notes and topics

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

Functional capabilities of chat protocol

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

File Transfer API for Agent Desktop

Warning

This content has been moved to the *Chat Server Administration Guide* as of October 29, 2018 and, as such, the Chat Server content in this guide will no longer be maintained.

SMS Server Administration

This section provides information for SMS Server administrators. In addition to the topics on this page, see also [Supported SMPP v3.4 Operations](#).

SMS Server Handles Empty Messages

SMS provider sometimes delivers empty messages (no text, no payload). These messages must be processed by an agent. To send empty messages to an agent for processing, enter text for the value of the `x-smpp-empty-message` configuration option. This text will be delivered to an agent as a content of the original empty message.

Important

The text you specify for the `x-smpp-empty-message` configuration option can optionally be an empty string.

SMS Server Masks Sensitive Data in Logs

To mask sensitive data, perform the following two steps:

1. Set the `logging-filter-active` configuration option in the settings section to `true`, to activate SMS Server specific filtering.
2. Apply the standard log filtering. Standard log filtering covers Key-Value pairs in User Data of ESP requests, which can contain sensitive information. Use `_smsText`, `Subject`, `FromAddr`, `PhoneNumber`, and `_smsDestNumber` Key-Value pairs for filtering. To filter out information in the target Key-Value pair, in the `log-filter-data` section create an option with the same name as the Key-Value pair (for example, `Subject`) and set its value to `hide` or `skip`.

SMS Server Processes Extra Parameters for PDU

Some SMPP commands (or PDU – Protocol Data Unit) contain a set of optional parameters. Some SMS providers require these optional parameters to correctly process SMPP protocol. SMS Server supports this functionality – you can define PDU's optional parameters on two levels:

- As a parameter of ESP request – Inside ESP request in a strategy, optional parameters are defined by the Key-Value pair `extraopt` in Extra optional parameters of `Send SMS Out` block.
- As a server option – In the server, optional parameters are defined by the value of the `x-smpp-extraopt` option in the `channel-any_name_for_sms` section.

If optional parameters are defined in the ESP request, they are used in PDU. If optional parameters are not defined in the ESP request, they are taken from server option. If optional parameters are not defined in either the ESP request or the server option, PDU is formed without optional parameters.

Whether defined in the ESP request or the server option, optional parameters are defined by the string of the format, as described below.

In SMPP, optional parameters are defined as a triplet (tag, length, value) called the TLV value. SMS Server implements the following format of the string, representing optional parameters:

```
tlvItems:
[
  { tag:<tag value>, typ:<value type (i.e. byte or int or octets or short or strnz or strz)
>, val:<value> }
  . . . MORE TLV SPECIFICATIONS SEPARATED BY COMMAS
]
```

A parameter's type defines encoding and data size placed in PDU:

- byte is coded as 1-byte integer binary value
- int is coded as 4-bytes integer binary value
- octets is coded as a sequence of bytes, specified as a sequence of hex values
- short is coded as 2-bytes integer binary value
- strnz is coded as ASCII sequence with a length, defined by the string content without adding a terminating zero byte
- strz is coded as ASCII sequence ('CString' as in SMPP specification) with a length, defined by the string content with terminating zero byte added

Example:

```
{tlvItems:[{tag:5121, typ:octets, val:0011aacc00}, {tag:5122, typ:strz, val:abc123},
{tag:5130, typ:strnz, val:abc123}, {tag:5131}]}
```

Produces:

```
(extraopt: (oct: (tlv: 5121) 0011aacc00) (oct: (tlv: 5122) 61626331323300) (oct:
(tlv: 5130) 616263313233) (oct: (tlv: 5131) ) ) )
```

SMS Server Supports Message Throttling by Configurable Rate

The SMS service provider might impose limits on the frequency with which they accept SNMP messages.

SMS Server has configurable options that define parameters of communication with SMSC, including an option to control the maximum rate at which messages are sent to the SMS Center:

- x-smpp-sar-max-segments
- x-smpp-submit-max-rate
- x-smpp-submit-window-size
- x-smpp-response-max-waiting-time

SMS Server Supports SMPP v3.4 Operations

SMS Server supports the following SMPP v3.4 operations:

- BIND_TRANSCEIVER
BIND_TRANSCEIVER_RESP

The purpose of the SMPP bind operation is to register an instance of an ESME (External Short Messaging Entity) with the SMSC (Short Message Service Center) system and request an SMPP session over this network connection for the submission or delivery of messages.

- UNBIND
UNBIND_RESP

The purpose of the SMPP unbind operation is to deregister an instance of an ESME from the SMSC and inform the SMSC that the ESME no longer wishes to use this network connection for the submission or delivery of messages.

- SUBMIT_SM
SUBMIT_SM_RESP

This operation is used by an ESME to submit a short message to the SMSC for onward transmission to a specified short message entity (SME).

- DELIVER_SM
DELIVER_SM_RESP

DELIVER_SM is issued by the SMSC to send a message to an ESME. Using this command, the SMSC may route a short message to the ESME for delivery.

- ENQUIRE_LINK
ENQUIRE_LINK_RESP

This message can be sent by either the ESME or SMSC and is used to provide a confidence check on the communication path between an ESME and an SMSC.

The protocol referred to in this section is described in [Short Message Peer to Peer Protocol Specification v3.4, 12-Oct-1999 Issue 1.2](#)

Digital Administration

Refer to the [Digital Administration](#) product page for the latest information.

UTF-8 in Classification and Training Servers

Classification and Training Servers implement UTF-8 support as follows:

1. IF the server finds the JVM property `-Dgenesys.mcr.stdserverex.file.encoding=UTF-8`
2. THEN the server configures all connections with Genesys servers and clients as UTF-8.
The server does not change any JVM settings. To output non-ASCII characters to log files correctly, you must manually set the JVM property `-Dfile.encoding=UTF-8`.
3. IF the JVM property `-Dgenesys.mcr.stdserverex.file.encoding` is not found, and the JVM option `-Dfile.encoding=UTF-8` is found
4. THEN the server and the whole JVM work in UTF-8 mode, including all connections with Genesys servers and clients. The server does not change any JVM settings
5. ELSE IF the server is informed that Configuration Server uses UTF-8
6. THEN the server configures all connections with Genesys servers and clients as UTF-8.
This does not change any JVM settings. To output non-ASCII characters to log files correctly, you must manually set the JVM property `-Dfile.encoding=UTF-8`.
7. ELSE the server does nothing.

Note also the following:

- To display data encoded as UTF-8 on Windows, you must adjust the Windows console as follows:
 - Set it to a non-raster font capable of showing non-ASCII symbols.
 - Set the console's code page with the command `chcp 65001`.
- If an application uses Platform SDK to connect with a Genesys server, and the application and the server have different localization settings, some [manual adjustment of the configuration](#) may be required.

eServices Manager

eServices Manager allows users to create and store Knowledge Management objects (categories, screening rules, standard responses, and so on) in a multi-tiered hierarchical structure. At the time of release, Composer and Orchestration Server do not support this new hierarchical structure. For this reason, customers using release 8.1.4. or earlier of Composer and Orchestration Server are advised to **not** design their screening rules in multilevel hierarchy trees.

Knowledge Manager (legacy)

This section deals with the Knowledge Manager component that is a standalone Windows application.

Basic Limitations

For Knowledge Manager, observe the following limitations:

- Categories: 3,500 categories
- Standard responses: 50 per category
- Attachments: 20 per standard response, 5 MB per attachment
- Field codes: 1,000
- Screening rules: 1,000
- Training objects: 200,000 e-mails, 20 KB per e-mail, 510 B for each e-mail's subject field

Screen Resolution

For Knowledge Manager to operate correctly, you must set a minimum screen resolution of 1280 x 1020.

Memory Allocation

You can adjust the memory size that Java allocates for Knowledge Manager processes by using the parameter `-Xmx1000m` in the following line in the `.bat` file:

```
start "Knowledge Manager" /b "%GES_HOME%\jre\bin\javaw"  
-Xmx1000m -classpath %CLASSPATH% -Djava.security.manager  
-Djava.security.policy=. \java.policy Genesys.iknow.manager.TM_start %*
```

`-Xmx1000m` means that 1,000 MB is allocated for Knowledge Manager; changing this number changes the allocation. The following considerations bear on adjusting this parameter.

- In some cases, Knowledge Manager does not work when you attempt to launch it from a machine that has a remote connection to the host of Knowledge Manager. As a workaround, lower the value of `-Xmx1000m` to `-Xmx512m`. In the unlikely event that this does not work, try a further decrease to `-Xmx256m`.
- You may want to adjust this parameter for better performance with large training objects (see [See Large Training Objects](#)), or before importing or exporting large files. For DB2 and Oracle, see also the recommendations in [Adjusting Database Configuration](#) below.

However, if this parameter is too low, it may impose limits on Knowledge Manager lower than those listed in [Basic Limitations](#) above. If so, you can consider increasing this parameter.

There is a similar issue with [UCS](#).

Adjusting Database Configuration

To prevent problems when using Knowledge Manager to import or export very large files, Genesys has the following recommendations about database configuration.

- For DB2, do as follows:
 1. In the DB2 Control Center, select System > Instance > Databases.
 2. Select the database desired.
 3. Right-click the desired database.
 4. In the resulting shortcut menu, select Configure.
 5. In the resulting dialog box, select Logging.
 6. Increase the number of files and/or file size.
- For Oracle, use Enterprise Manager to increase the number of rollback segments. Refer to Oracle documentation for details.
- For Microsoft SQL, no special configuration is needed.

Managing Sensitive Data

Interactions may contain data that should not be stored or even displayed; examples are credit card numbers, phone numbers, and bank account numbers. Genesys provides several ways of managing this type of data.

To manage sensitive data, you may want to:

- Replace it with something, such as a string of repeated meaningless characters like ****, or a message like <account number omitted>.
- Send an alert message when the system encounters sensitive data.
- Or both.

How do I manage sensitive data

Procedures for managing sensitive data differ somewhat according to the media of the interaction, but in all cases you use **rules** that scan interactions for sensitive data and then take some action.

- E-mail—Use **Privacy Manager** to choose default rules that Genesys provides or create your own rules, then use routing strategies to implement the rules.
- Chat—Use **Chat Server options** to configure whether and how Chat Server looks for sensitive data and what it does with it. As for rules, you can:
 - Use the default rules with no further configuration.
 - Use Privacy Manager to create your own rules.

Chat Server performs sensitive data management on its own, with no need for anything specific in a routing strategy.

- Other media—Use **routing strategies** to implement default rules that Genesys provides.

Rules

The rules use **regular expressions** to look for sensitive data and then do something. What they can do:

- Replace the data with something unrevealing
- Send a notification that sensitive data was found

Other points to know about rules:

- A rule contains a regular expression (RegEx) as well as other attributes such as the name of the rule, its

priority relative to other rules, and the pattern to be used in replacing the sensitive data.

- Rules come in *groups*. In Privacy Manager, rules are grouped according to media: chat rules and email rules.
- Regular expressions must use the same syntax and semantics as defined for `java.util.regex`. However, for chat it must also comply with [Perl 5 defined syntax and semantics](#).

You can use Privacy Manager to write your own rules and test them, but Genesys also provides hard-coded rules that use the following regular expressions:

Name	Regular Expression
Credit Card (Visa and MasterCard only)	<code>(?>^(?<=[\s[:alpha:]](,,:;!'"`)))(?>4\d{3} 5[1-5]\d{2} 6011 622[1-9] 64[4-9]\d 65\d{2})[-]?\d{4}[-]?\d{4}[-.]?\d{4}(?>\$ (?=[\s[:alpha:]](,,:;!'"`)))</code>
Phone Number (North America)	<code>(?>^(?<=[\s[:alpha:]](,,:;!'"`)))(?:\+?1[-.])?(?:\([2-9][0-9]{2}\)?[-.])?[2-9][0-9]{2}[-.]?[0-9]{4}(?>\$ (?=[\s[:alpha:]](,,:;!'"`)))</code>
SSN (Social Security Number - U.S. only)	<code>(?>^(?<=[\s[:alpha:]](,,:;!'"`)))(?!000 666 9)\d{3}[-.]?(?!00)\d{2}[-.]?(?!0000)\d{4}(?>\$ (?=[\s[:alpha:]](,,:;!'"`)))</code>

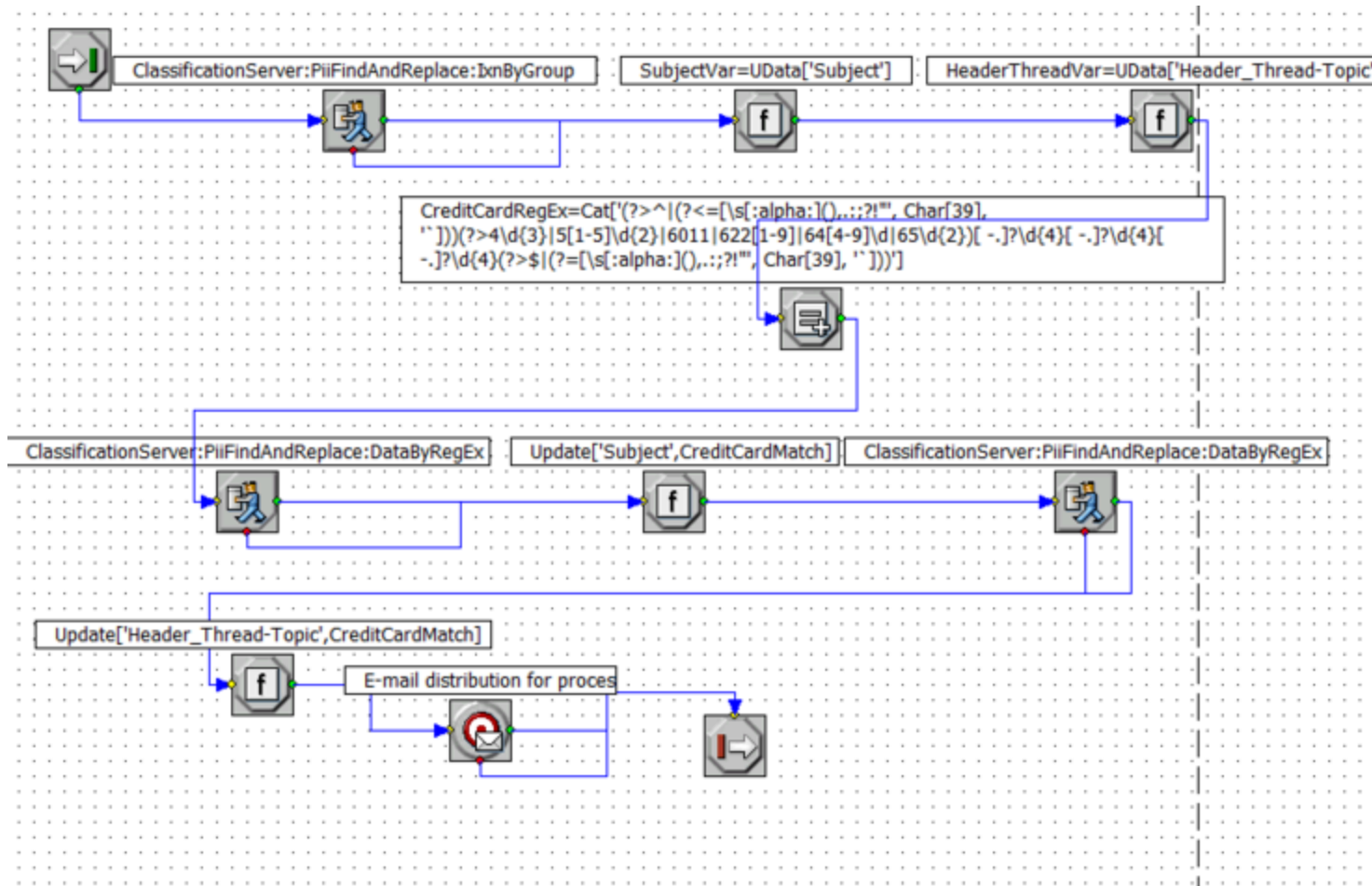
Routing Strategies

For channels other than Chat, you must use [Composer](#) or Interaction Routing Designer (see the [Universal Routing](#) documentation) to create strategies (or modify existing ones) that include an External Service object that calls one of the following methods:

- `IxnByGroup`—This method specifies an interaction in the UCS database and the group of rules to apply to it. Its parameters are listed [below](#).
- `DataByRegex`—This method extracts the text to be screened from the interaction as it passes through the strategy and the regular expression to apply to the text. Use it when you do not want to (or cannot) retrieve the interaction from the UCS database. In addition to the External Service object, strategies using this method must include some strategy object that extracts content from the user data and puts it in a variable which it passes to the External Service object. The parameters of this method are listed [below](#).

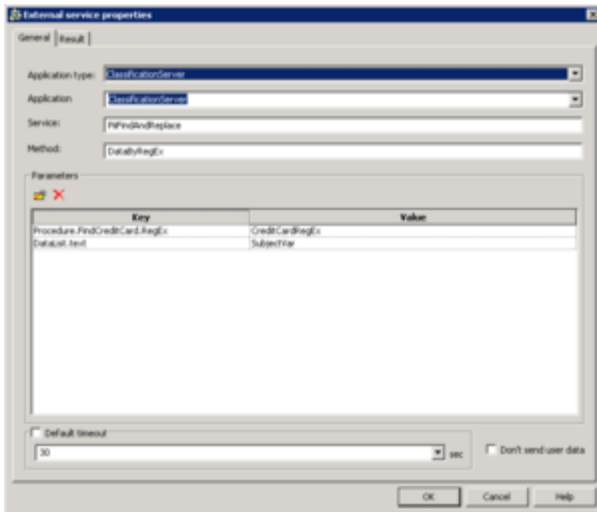
Sample Strategy

The following strategy illustrates the use of both methods on an email interaction:

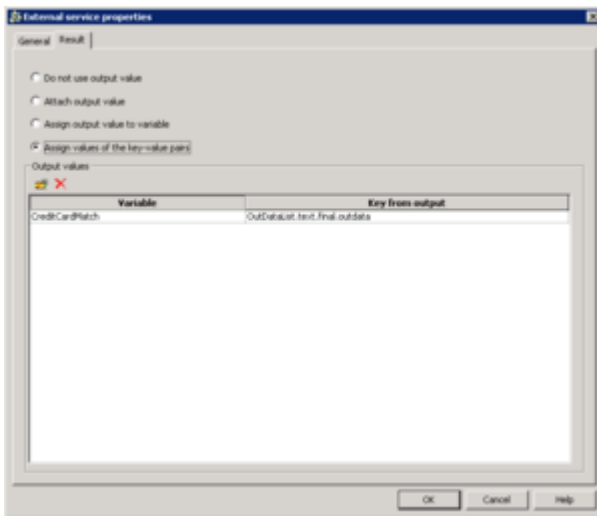


Strategy Using IxnByGroup and DataByRegex

1. In the first **External Service** object, IxnByGroup looks at the interaction in the UCS database and scans its entire content: Subject, Header, Body. It also updates the content of the interaction as stored in UCS, replacing any sensitive data that it finds with strings of * (asterisks). However, IxnByGroup does not affect the interaction's User Data, which contains attributes, such as Subject and various headers, that might also contain sensitive data. For that we must use DataByRegex.
2. Two Function objects retrieve the content of the Subject and Thread-Topic.
3. A Multi-Assign object creates a variable CreditCardRegex and assigns it a value consisting of a regular expression that finds credit card numbers.
4. In the second **External Service** object, DataByRegex scans the content of the Subject field.



External Service, General Tab (click to enlarge)



External Service, Result Tab (click to enlarge)

5. The following Function object updates the interaction (in the Interaction Server database), substituting * for the found data.
6. The third **External Service** object does the same for the Thread-Topic field.
7. When the interaction is terminated, the User Data attributes are also updated in the UCS database.

IxnByGroup Parameters

Parameter	Type	Description	Mandatory?	Default Value
Group	String	ID of the rule group to be applied. Either Group or GroupName may be specified, but	N	No default value

Parameter	Type	Description	Mandatory?	Default Value
		not both. If both are specified an Error is generated. If neither is specified, the predefined Email group is used.		
GroupName	String	Name of the rule group to be applied. Either Group or GroupName may be specified, but not both. If both are specified an Error is generated. If neither is specified, the predefined Email group is used.	N	Email
IxnAccessSpec	List	Specifies which parts of interaction stored in UCS should be processed, and other parameters needed for Ixn. Ucs Access Provider. This string is passed to the Provider and is used by the Provider exclusively. The string has the following form: key:value=<part>:<operation>, where <i>part</i> can be Subject, Header, Text (body), StructuredText, Content (MIME content), or _EmailAll (all fields) <i>operation</i> can be check (the modified part of the interaction is not written back to UCS) or update (the modified part of the interaction is written back to UCS).	N	key:value=_AllEmail:update
IxnList	String	List of IDs of interactions stored in UCS, separated by the pipe character (). If	N	No default value

Parameter	Type	Description	Mandatory?	Default Value
		absent, the Interaction ID is taken from user data.		
ProcedureOpt	String	<p>Sets the output type of the procedure:</p> <ul style="list-style-type: none"> final—only final processed data is placed in the result trace—full output with results of all intermediate procedure steps, including positions, is placed in the result. 	N	final

DataByRegex Parameters

Parameter	Type	Description	Mandatory?	Default Value
DataList	List	Specifies the data portions to process: a list of key-value pairs, where the key is the reference ID of this data portion and the value is a string specifying the data portion to process.	Y	No default value
Procedure	List of K-V pairs	<p>Describes find and replace procedure by direct explicit specifications of its steps.</p> <ul style="list-style-type: none"> Key (String)—Reference ID of this step of the procedure Value: (List of key-value 	Y	No default value

Parameter	Type	Description	Mandatory?	Default Value
		pairs)—Specification of this step of the procedure, as listed in "Values of Procedure " below.		
ProcedureOpt	String	<p>Sets the output type of the procedure:</p> <ul style="list-style-type: none"> final—only final processed data is placed in the result trace—full output with results of all intermediate procedure steps, including positions, is placed in the result. 	N	final

Values of **Procedure**

Key	Type	Description	Default Value
TheOrder (optional)	Integer	Specifies the order of this part of the procedure. If Procedure contains only one step then TheOrder can be omitted. Otherwise TheOrder must be specified for each step of the procedure, and each step must have a different value.	No default value
RegEx (mandatory)	String	Regular expression used to process the data	No default value
ReplacementPattern (optional)	String or K-V list	Replacement pattern applied in data processing.	See embedded table to left

Key	Type	Description	Default Value
"name"	String	Name of the replacement pattern	Empty string
"type"	String	Type of replacement pattern: <ul style="list-style-type: none"> "standard"—POSIX type, with named and numbered capturing groups "genesys"—as specified in the "spec" attribute 	"genesys"
"spec"	String	Specification of the replacement pattern if "type" = "genesys": <ul style="list-style-type: none"> "none"—Replace nothing "replace-digits-0"—Replace all characters in the found text "replace-digits-<N>"—Replace 	"replace-digits-0"

Key	Type	Description	Default Value
		only digits in the found text, leaving the <N> rightmost digits, where <N> is a non-negative integer.	
	String	Specifies a character used to replace all (asterisk) characters in the found text	

Response

The response to the above methods is Event3rdServerResponse, which has the following parameters:

Key	Type	Description	Default Value
OutDataList	List of lists	Key is the reference ID of the original data portion: the interaction ID or the reference ID of the data portion in the DataList parameter of the request. The value is as follows	No default value

Key	Type	Description	Default Value
		The key is "outdata" and the value is a string No default value	
"final"	List of key-value pairs	The key is "outdata" and the value is a string No default value of the final result of processing this data portion.	
Reference ID of a step of the procedure (the key in the request's Procedure key-value list).	List of key-value pairs	String: the result "outdata" of processing this step	
		List: positions of text found in No "posfound" this default processing step; see description below.	
These data elements are created only when ProcedureOpt = trace.		List: positions of texts "posinserted" in place of found	

Key	Type	Description	Default Value																
		<table border="1"> <thead> <tr> <th>Key</th> <th>Type</th> <th>Description</th> <th>Default Value</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td> <table border="1"> <thead> <tr> <th>Key</th> <th>Type</th> <th>Description</th> <th>Default Value</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td>text; see description below.</td> <td></td> </tr> </tbody> </table> </td> <td></td> </tr> </tbody> </table>	Key	Type	Description	Default Value			<table border="1"> <thead> <tr> <th>Key</th> <th>Type</th> <th>Description</th> <th>Default Value</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td>text; see description below.</td> <td></td> </tr> </tbody> </table>	Key	Type	Description	Default Value			text; see description below.			
Key	Type	Description	Default Value																
		<table border="1"> <thead> <tr> <th>Key</th> <th>Type</th> <th>Description</th> <th>Default Value</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td>text; see description below.</td> <td></td> </tr> </tbody> </table>	Key	Type	Description	Default Value			text; see description below.										
Key	Type	Description	Default Value																
		text; see description below.																	

Values of "posfound" and "poschanged"

Key	Type	Value
"start"	String	Starting position. The first character in a string is numbered 0.
"end"	String	Ending position.

Masking Sensitive Data

Logs and chat transcripts may contain sensitive data such as credit card numbers, phone numbers, Social Security numbers, and so on. You can omit this data from logs and mask it in transcripts by properly configuring Chat Server and UCS.

Tip

In addition to the method of configuration described on this page, you can also use [Privacy Manager](#), a plugin for Genesys Administrator Extension (GAX), to create, select, and activate rules that handle sensitive data.

UCS

Some configuration of UCS is required for all channels. The following table summarizes what is needed for each channel.

Channel	Section	Option	Value
All	log	log4j.logger.ucs.svc.indexingarn	
All	settings (of UCS DAP)	interpret-prepared-statements	false
Chat	log-filter-data	StructuredText	hide
Chat	log-filter-data	Transcript	hide

Chat Channel

Logs

- In the **[settings]** section, set message-log-print-size to 0. This means that logs do not show the messages sent between chat participants. Where a message occurs, the log shows [truncated from size=x], where x is the number of characters in the suppressed message.
- In the **[log-filter-data]** section,
 - Set StructuredText to hide so that logs will omit the transcript that UCS sends to Chat Server.
 - Set Transcript to hide so that logs will omit the transcript that Chat Server sends to UCS.

Transcripts

	Name	Regular Expression
Credit card	GCTI_CreditCards	<p>8.5.0 version only:</p> <pre>(?>^(?<=[\s:\alpha:()...;?!"'\`])(?>4\d{3}[5[1-5]\d{2} 6011 622[1-9] 64[4-9]\d{65}\d{2})[-]?\d{4}[-]?\d{4}[-]?\d{4}(?>\$ (?=[\s:\alpha:()...;?!"'\`]))</pre> <p>8.5.1 version only:</p> <pre>(?>^(?<=[\s:\alpha:()...;?!"'\`])(?>4\d{3}[5[1-5]\d{2} 6011 622[1-9] 64[4-9]\d{65}\d{2})[-]?\d{4}[-]?\d{4}[-]?\d{4}(?>\$ (?=[\s:\alpha:()...;?!"'\`]))</pre>
Social Security Number	GCTI_SSN	<pre>(?>^(?<=[\s:\alpha:()...;?!"'\`])(?!000 666 9)\d{3}(?!00)\d{2}[-]?(?!0000)\d{4}(?>\$ (?=[\s:\alpha:()...;?!"'\`]))</pre>
Phone number using the North American Numbering Plan	GCTI_PhoneNANPA	<pre>(?>^(?<=[\s:\alpha:()...;?!"'\`])(?:\+?1[- .])?(?:\([2-9][0-9]{2}\)\)?[- .])?[2-9][0-9]{2}[- .]?[0-9]{4}(?>\$ (?=[\s:\alpha:()...;?!"'\`]))</pre>

Typing Preview

Typing preview allows an agent to see text that a customer types before the text is submitted to the chat session. You can have Chat Server mask all digits in the typing preview by setting the typing-preview (called transcript-cleanup-typing before release 8.5.103) option to a value other than none. Chat Server then replaces all digits in the typing preview with the character specified by default-repchar (called transcript-cleanup-mask before release 8.5.103).

General Recommendations

This section presents some recommendations for monitoring and adjusting your eServices configuration.

In addition to the items on this page, see also the following topics:

- [Database Performance](#)
- [Unicode Character Support](#)
- [Classification Server](#)
- [SMS Server](#)
- [Security](#)

Parameters to Check

Check that the following parameters do not significantly exceed their average values:

- Memory usage
- CPU load
- Number of handles for eServices-related processes (with Windows operating system)

Loading on Application Servers

Monitor the loading on application servers (Classification Server, E-mail Server). If application servers are being overloaded, do one or both of the following:

- For all routing strategies that process interactions with no agent involvement, adjust the limit on the number of interactions that Interaction Server can submit to Universal Routing Server (URS). You can set this limit for a strategy using the `max-submitted-interactions` option. See "Interaction Server Options" in Chapter 2 of the [eServices 8.1 Reference Manual](#).
- Add instances of the required application server on other hosts.

Database Performance

This page provides general recommendations on running Microsoft SQL.

Further information on improving the performance of the **Interaction Server database** is also available.

Microsoft SQL 2000

- In general, patch up to Service Pack 4.
- If Microsoft SQL is running on a machine with over 2 GB of RAM, use Windows' AWE (Address Windowing Extensions) mode. To avoid performance degradation, patch Microsoft SQL according to Microsoft's recommendation "FIX: Not all memory is available when AWE is enabled on a computer that is running a 32-bit version of SQL Server 2000 SP4" (see <http://support.microsoft.com/?kbid=899761>). This patch brings Microsoft SQL to version 8.00.2040.

Microsoft SQL 2005

There is an issue that occurs with Microsoft SQL 2005: when the database is very large (on the order of one million interactions), there are periodically exceptions in the `Stat` service, and CPU activity rises to 100%.

To avoid this issue, configure the UCS DAP as follows:

1. Create a settings section.
2. In this section, create an option called `prepare` and set its value to `false`.

This DAP configuration applies to Microsoft SQL 2005 only; configuring the DAP in this way with Microsoft SQL 2000 degrades performance.

Unicode Character Support

Although UCS supports Unicode character sets, other components of eServices and of the Genesys suite do not (in particular, URS and Interaction Server 8.5.0 and earlier—however, Interaction Server 8.5.1 and later does support UTF-8 on Windows deployments). This means that interactions that use a Unicode character set may be corrupted. Specifically, what may be corrupted is any part of the interaction's data that is handled by Interaction Server or URS (or any other component that does not support Unicode). This includes attributes such as Subject, FirstName, and LastName. It does not include the body of the interaction, which is handled by UCS only.

The following scenario provides an example of how this corruption can happen:

1. URS processes an interaction that includes Unicode user data, such as Subject. Because URS does not support Unicode, the Subject and other user data is corrupted.
2. UCS receives `RequestStopProcessing`, either from URS or the agent desktop.
3. UCS saves the interaction's user data (this is done in case the user data has changed during processing), copying certain properties from the user data into the corresponding fields of its `Interaction` table.
This user data includes the interaction's subject, the value of which is copied into the `Subject` attribute. But the user data was corrupted during processing by URS, so the corrupted data is now stored in the UCS database.
4. If the corrupted subject data is used to compose another e-mail (such as reply or redirect), the subject of the new e-mail is also corrupted.

As a workaround for this scenario, you can modify the applicable strategy so that it deletes the Subject user data before it issues `RequestStopProcessing`.

Classification Server

This section provides information for administrators regarding Classification Server.

Modifying any of the following may have repercussions elsewhere in the system:

- Categories
- Standard responses
- Field codes
- Screening rules

If you modify any of these objects, it would be prudent to check any compiled strategies that use the following:

- Acknowledgment
- Attach Categories
- Autoresponse
- Chat Transcript
- Classify
- Classify Switch
- CreateEmailOut
- CreateSMS
- Forward
- Multi Screen
- Screen

Perform this check by recompiling the strategies in question. If this is not possible, monitor the Classification Server log for errors related to screening rules and UCS logs for errors related to rendering of standard responses.

Security

Genesys makes the following security recommendations for deploying eServices:

- Put Web API Server in the DMZ.
- Put all other eServices components in the internal network.
- Open ports in the firewall between the DMZ and the internal network to allow Web API Server to connect with other eServices components. The following table lists each component and the port to open.

Port Types in Firewall

Server	Port
Configuration Server	Default port on Server Info tab
Message Server	Default port on Server Info tab
Solution Control Server	Default port on Server Info tab
Interaction Server	Default port on Server Info tab
Chat Server	Port specified by the <code>webapi-port</code> option in the settings section. If not specified, default port on Server Info tab.
E-mail Server	Port specified by the <code>webapi-port</code> option in the settings section
Stat Server	Default port on Server Info tab
Co-Browsing Server	HTTPS
UCS	Port specified by the <code>ucsapi</code> option in the ports section

- Open a port in the firewall to allow Solution Control Server to connect to the Local Control Agent (LCA) located on the host of Web API Server.
- Open ports in the firewall to allow SMS Server to connect to the SMSCs specified in the SMS Server's configuration