eServices Administrator's Guide

Managing Sensitive Data

12/19/2025

Contents

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 date.

- 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)	(?>^|(?<=[\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:]](),.:;?!"'`]))
Phone Number (North America)	(?>^|(?<=[\s[:alpha:]](),.:;?!"'`]))(?:\+?1[-. ]?)?(?:\(?[2-9][0-9]{2}\)?[-. ]?)?[2-9][0-9]{2}[-. ]?[0-9]{4}(?>$|(?=[\s[:alpha:]](),.:;?!"'`]))
SSN (Social Security Number - U.S. only)	(?>^|(?<=[\s[:alpha:]](),.:;?!"'`]))(?!000|666|9)\d{3}[-.]?(?!00)\d{2}[-.]?(?!0000)\d{4}(?>$|(?=[\s[:alpha:]](),.:;?!"'`]))

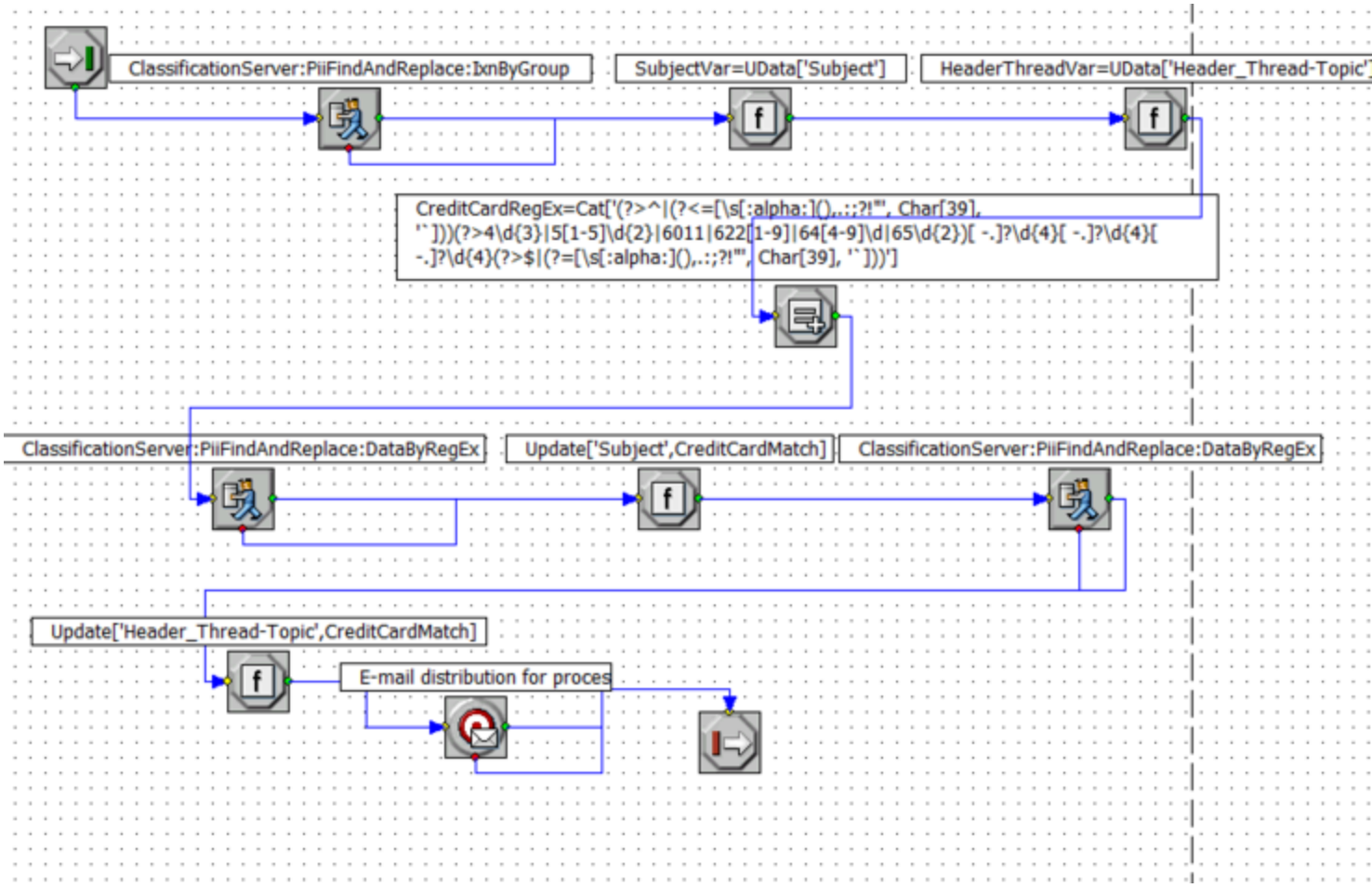
Routing Strategies

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.

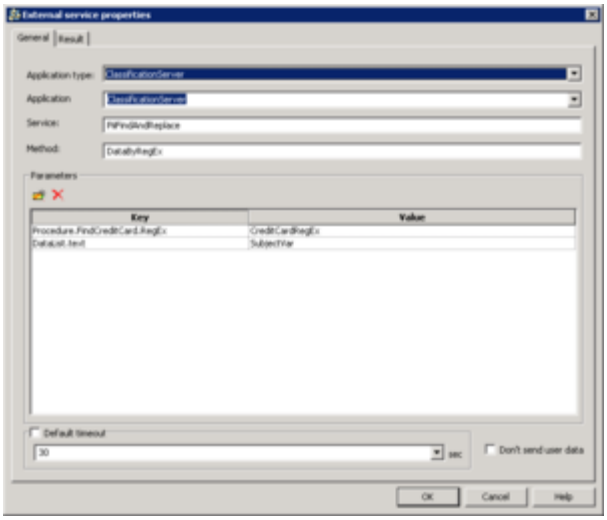
Sample Strategy

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

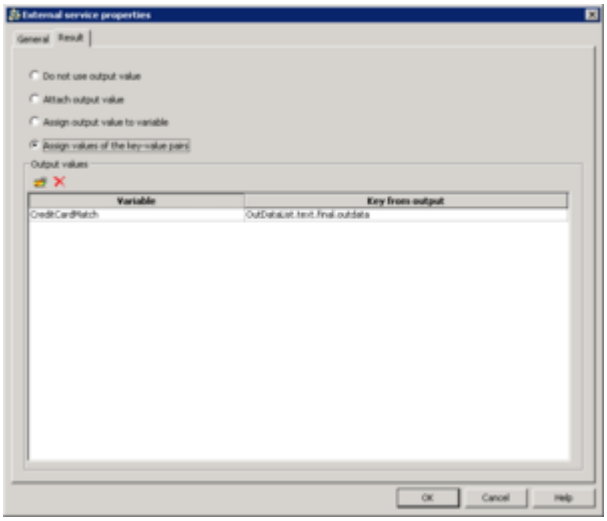


Strategy Using IxnByGroup and DataByRegEx

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 *part* can be `Subject`, `Header`, `Text` (body), `StructuredText`, `Content` (MIME content), or `_EmailAll` (all fields) *operation* 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	Sets the output type of the procedure: • `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	Describes find and replace procedure by direct explicit specifications of its steps. • 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	Sets the output type of the procedure: • `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
		<table><tr><th>Key</th><th>Type</th><th>Description</th><th>Default Value</th></tr><tr><td>"name"</td><td>String</td><td>Name of the replacement pattern</td><td>Empty string</td></tr><tr><td>"type"</td><td>String</td><td>Type of replacement pattern: • "standard"—POSIX type, with named and numbered capturing groups • "genesys"—as specified in the "spec" attribute</td><td>"genesys"</td></tr><tr><td>"spec"</td><td>String</td><td>Specification of the replacement pattern if "type" = "genesys": • "none"—Replace nothing • "replace-all"—Replace all characters in the found text • "replace-digits-<N>"—Replace</td><td>"replace-digits-0"</td></tr></table>	

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

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

Key	Type	Description	Default Value
		<table><tr><th>Key</th><th>Type</th><th>Description</th><th>Default Value</th></tr><tr><td></td><td></td><td><table><tr><th>Key</th><th>Value</th></tr></table></td><td>text; see description below.</td></tr></table>	

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.