Orchestration Server Deployment Guide

Hiding Sensitive Data

12/18/2025

Contents

Hiding Sensitive Data

Starting with Release 8.1.400.30, Orchestration Server can selectively hide sensitive data in logs when the printing of such sensitive data is not explicitly requested by an SCXML strategy. Because Orchestration Server prints Event as a whole, sensitive data may appear in ORS logs in the following scenarios.

Type of Logging	Supported?	Source for ORS Configuration Information
Headers of HTTP messages triggered by the `<session:fetch>` action.	Supported starting with ORS Release 8.1.401.03.	See Hiding Sensitive Data below.
Create/Update `ORSURS` requests for multimedia interactions.	Supported starting with ORS Release 8.1.400.30.	See Hiding Sensitive Data below.
Incoming Web requests, processed by WebFM and outgoing Web requests from WebFM.	Supported starting with ORS Release 8.1.400.30. **Important** Incoming/outgoing Web requests may be printed as-is by underlying library (in addition to logging in ORS FMWeb debug logging segment), and that log output cannot be filtered via options, described in this topic. If it is the case, that low-level logging may be suppressed via the ORS option `log\x-server-http-trace-level`, value less than 3.	See Hiding Sensitive Data below.
Incoming TEvents/Requests from/ to T-Server/SIP Server.	Yes. ORS and other Genesys servers already support this functionality.	See the Genesys Security Deployment Guide. Also see the log-filter-data section of the *Platform SDK Developer Guide*.

The configuration information below applies to hiding sensitive data in `ORSURS` and `WebFM` requests.

Hiding Sensitive Data

To hide ORS log data that may be considered as completely or partially sensitive in `ORSURS` requests and Web requests (request/response URL, headers and body), ORS implements the usage of regular expressions. A regular expression is a pattern describing a certain amount of text. The name "regular expression" is frequently abbreviated to "regex" as described in the option below.

Before printing sensitive data in logs, ORS verifies that string data against all regular expressions defined in the `ors-regex-<name>` option. The portion of a string that satisfies a regular expression will be replaced with “****” symbols or tagged. When filtering and/or tagging data in logs, only the `hide` and `tag` capabilities are supported as described under `<filter-type>` below.

A new option supports the hiding of sensitive data in `ORSURS` request and Web request logging.

ors-regex-<name>[;<filter-type>]

- For `<name>`, specify a descriptive name describing the rule with the intent of having different option names. Note that `ors-regex` is case-sensitive.
- For `<filter-type>`, specify the way of hiding information in the log, when regex (regular expression) finds a match. See `<filter-type>` in the table below.

Object: ORS Application object

Option section: `log-filter-data`

Default value: No default value

Valid values: Any valid Perl-compatible regular expression (`regex`).

Value changes: Take effect immediately.

Use this option to define regular expressions for hiding/tagging part of a URL, the header or body of a Web request, or the content of an `ORSURS` request.

<filter-type>

Valid Value	filter-type Description
hide	Symbols, matched by a regular expression, will be replaced by asterisks.
tag	**tag[(<tag-prefix>,<tag-postfix>)]** The substring, matched by a regular expression, will be tagged with the prefix specified by <tag-prefix> and the postfix specified by <tag-postfix>. If the two parameters are not specified, the default tags <# and #> are used as prefix and postfix, respectively. To use the default tags, you can use any of the following values: • tag • tag() • tag(,) To define your own tags, replace the two parameters in the value with your tags. Your own tag can be any string up to 16 characters in length; any string longer than that will be truncated. If the string includes a blank space or any of the characters , (comma), (, or) as start and stop characters, they will not be counted as part of the length of the string. For information about how to use the hide and tag filter types, refer to the description of `default-filter-type` option in the Framework 8.5 Configuration Options Reference Manual. See Filtering and/or Tagging Data in Logs.

ORS 8.1.400.31 introduced option, `filter-eval-expr`, which works with `METRIC:eval_expr` and Debug Logging Segmentation (see option `log-trace-segments`).

filter-eval-expr

Option section: `orchestration`
Configuration object: ORS `Application` object
Default value: `false`
Valid values: `true` or `false`
Value changes: Immediately upon notification

This option enables hiding of sensitive data in expression and result fields of the `eval_expr` log metric. If set to `true`, ORS verifies that string data against the regular expression defined in the `ors-regex-<name>` option.

Beginning with release 8.1.400.91, ORS does not consider the `expression` and `result` fields of the `eval_expr` metric as independent while hiding sensitive data contained in them. If information considered sensitive is found in the expression field (by matching with the regular expression specified in the `ors-regex-<name>` option), the result field is hidden automatically.

Important

One of the reasons for auto-hiding the `result` field is that at times it is possible to decipher sensitive data contained in the result of a previous expression if a subsequent expression and its result are known, even if the result by itself might not look like it contains sensitive data. Users must consider this when formulating regular expressions.

The following sample illustrates the above change:

In this sample, let us assume 123 is sensitive data.

Original:

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var aaa=123;' …
result='123' />
```

Processed and hidden:

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var aaa=****;' …
result='****' />
```

A subsequent metric could implicitly expose the hidden sensitive data:

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var bbb=aaa+100;'
… result='223' />
```

As you know the expression bbb=aaa+`100` and its result 223, you can easily infer the value of the aaa variable.

Regular expression matching is quite a costly operation in terms of performance. So, you should provide regular expressions as simple as possible but yet be able to hide what you want to hide. And their number should be minimal as well.

In reality, there could be multiple (as many as you need) variables and object properties holding sensitive data in your strategies. The following tactic is recommended as an example only without any legal obligations:

- All names of variables and object properties holding sensitive data should start with one common prefix that distinguishes them from others. For example, the prefix could be `sdhldr_` that stands for *sensitive data holder*.
- You can follow that with regular expressions that hide everything after the prefix up to the end of the statement or the expression field value, for example:
    - `(?<=sdhldr_)[^;]*`
    - `(?<=sdhldr_).*`

Now, returning to our sample illustration, the original metrics in case of using the first regex provided above:

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var
sdhldr_aaa=123;' … result='123' />

METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var
bbb=sdhldr_aaa+100;' … result='223' />
```

Will be securely treated as:

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var sdhldr_****;'
… result='****' />

METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var
bbb=sdhldr_****;' … result='****' />
```

Both the result fields will be hidden automatically as sensitive data is found in the related expression fields.

Headers of HTTP messages triggered by the `<session:fetch>` action

Beginning with version 8.1.401.03, the Hiding Sensitive Data feature is now applicable to logging of the headers of HTTP messages triggered by the `<session:fetch>` action.

To enable the feature, configure the `ors-regex-<name>` options in the `log-filter-data` section.

If the options or section are not configured, a default rule, `ors-regex-auth;hide=(?<=Authorization: ).*` is applied and the value of the Authorization header is hidden in the log. The default rule covers all options with the name, `ors-regex-auth;hide`, and value, `(?<=Authorization: ).*`.

Important

Default rules must be included in the ORS configuration template for new installations.

Samples

The samples below show complex regular expressions that can be used to hide substrings recognized as Social Security numbers, phone numbers, and credit card numbers.

Note: Line breaks in log message added for readability.

Regular expressions

[log-filter-data]

Credit Card:

```
(?>^|(?<=[\s[:alpha:](),.:;?!"'`=&]))(?>4\d{3}|5[1-5]\d{2}|6011|622[1-9]|64[4-9]\d|65\d{2})
([ -.]?|%20)\d{4}([ -.]?|%20)\d{4}([ -.]?|%20)\d{4}(?>$|(?=[\s[:alpha:](),.:;?!"'`=&]))
```

Matches only valid credit card numbers preceded/followed by any whitespace, Latin alphabetic, or any of (),.:;?!"'`=& characters, using positive lookbehind/lookahead.

4-digit groups can be delimited by a single space (or URL encoded as '%20'), period, or dash character. Make sure that this sample covers your cases before using it. Otherwise, modify the sample to fit your needs. Example:

```
http://example.com/send?cn=5555%205555%205555%205555&cvv=666
```

Phone Number using the North American Numbering Plan:

```
(?>^|(?<=[\s[:alpha:](),.:;?!"'`=&]))(?:(\+?|%2[Bb])1([ -.]?|%20))?(?:(\(|%28)?
[2-9][0-9]{2}(\)|%29)?([ -.]?|%20))?[2-9][0-9]{2}([ -.]?|%20)[0-9]{4}
(?>$|(?=[\s[:alpha:](),.:;?!"'`=&]))
```

Matches only valid phone numbers preceded/followed by any whitespace, Latin alphabetic, or any of (),.:;?!"'`=& characters, using positive lookbehind/lookahead.

An optional country calling code +1 (or %2B1 url encoded), 3-digit area code (can be surrounded with parentheses or URL encoded '%28'/'%29'), 3-digits exchange code and 4-digits subscriber number can be delimited by a single space (or URL encoded as '%20'), period, or dash character. Make sure that this sample covers your cases before using it. Otherwise, modify the sample to fit your needs. Example:

```
http://example.com/call?phone=%2B1%20%28415%29%20555%205555,
http://example.com/call?phone=%2B1(415)5555555,
http://example.com/call?phone=14155555555
```

Social Security Number:

```
(?>^|(?<=[\s[:alpha:](),.:;?!"'`=&]))(?!000|666|9)\d{3}([- ]?|%20)(?!00)\d{2}
([- ]?|%20)(?!0000)\d{4}(?>$|(?=[\s[:alpha:](),.:;?!"'`=&]))
```

Matches only valid Social Security Numbers preceded/followed by any whitespace, Latin alphabetic, or any of (),.:;?!"'`=& characters, using positive lookbehind/lookahead.

3-digits, 2-digits, and 4-digits groups can be delimited by a single space (or URL encoded as '%20') or the dash character. Make sure that this sample covers your cases before using it. Otherwise, modify the sample to fit your needs. Example: http://example.com/find?ssn=078%20051120&v=12

Hiding of Certain Properties in a JSON-Encoded Object

Initial Log Message:

```
05:35:30.049 {ORSURS:3} CreateVirtualCall: <<
refID        1
tenant        'Automation'
call        'da377034140aacad'
context
'{"MediaType":"email","UserData":{"ReceivedAt":"2015-07-30T12:35:28Z","InteractionId":"da377034140aacad","Media
"PlacedInQueueAt":"2015-07-30T12:35:28Z","InteractionType":"Inbound",
"InteractionSubType":"InboundNew","Queue":
"Qaart_EmailInboundQueueAutomation","SubmittedBy":
"QAART_Media_Server_email_Automation",
"SubmittedAt":"2015-07-30T12:35:28Z","
MovedToQueueAt":"2015-07-30T12:35:28Z","TenantId":101,
"InteractionState":0,
"IsOnline":0,"IsLocked":0,"next_kvlist":{"str":"some str","int":3},
"test_name":"eServiceTerminate"}}'
 <<06:25:11.525 {FMWeb:3} Send pending HTTP response:
Session ID = 9F7V6Q4L8P0OP7FUD0I8FDM4F4000001, Send ID = 2,
Result Code = Schedule_01_session_start_done,
Content = {"PhoneNumber":"+79211122333"}
```

Configuration:

```
[log-filter-data]
ors-regex-hide-testname=(?<="test_name":").*?(?=")
ors-regex-hide-mediatype;tag=(?<="MediaType":").*?(?=")
ors-regex-hide-part-of-phone-number=(?<="PhoneNumber":"\+7921).*?(?=")
```

Log Message After Hiding:

```
05:35:30.049 {ORSURS:3} CreateVirtualCall: <<
refID        1
tenant        'Automation'
call        'da377034140aacad'
context
'{"MediaType":"email","UserData":{"ReceivedAt":"2015-07-30T12:35:28Z","InteractionId":"da377034140aacad","Media
"<#email#>","PlacedInQueueAt":"2015-07-30T12:35:28Z","InteractionType":
"Inbound","InteractionSubType":
"InboundNew","Queue":"
Qaart_EmailInboundQueueAutomation","SubmittedBy":
"QAART_Media_Server_email_Automation",
"SubmittedAt":"2015-07-30T12:35:28Z",
"MovedToQueueAt":"2015-07-30T12:35:28Z",
"TenantId":101,"InteractionState":0,"IsOnline":0,
"IsLocked":0,"next_kvlist":{"str":"some str","int":3},"test_name":"****"}}'
 <<
06:25:11.525 {FMWeb:3} Send pending HTTP response:
Session ID = 9F7V6Q4L8P0OP7FUD0I8FDM4F4000001,
Send ID = 2, Result Code = Schedule_01_session_start_done,
Content = {“PhoneNumber”:”+7921****”}
```