



RRT3 Travel: Widget Documentation

Legal Notice

All rights reserved. The information contained in this document is confidential and may also be proprietary and trade secret. Without prior written approval from VRR no part of this document may be reproduced or transmitted in any form or by any means, including but not limited to electronic, mechanical, photocopying or recording or stored in any retrieval system of whatever nature. Use of any copyright notice does not imply unrestricted public access to any part of this document.

Copyright © 2021 VRR
Company Confidential

Verisk Risk Rating UK

Healix House, Esher Green
Esher
Surrey KT10 8AB
T +44 (0) 20 8481 7765

Verisk Risk Rating, Australasia

Level 17
324 Queen Street, Brisbane
Queensland, Australia 4000
T +61 (0) 7 3532 4555

verisk.com/insurance/products/risk-rating.html

Distribution

Organisation for Distribution	
Organisation	Organisation Address
Verisk Risk Rating UK Limited	Healix House, Esher Green, Esher, Surrey, KT10 8AB, United Kingdom
Verisk Risk Rating Australasia	Level 17, 324 Queen Street, Brisbane 4000

Document Control

This document is available in two forms, controlled and uncontrolled. The controlled variant is maintained electronically and accessed by authorised persons to the VRR folder in the shared drive. Uncontrolled variants are all other electronic and printed copies.

Title	RRT3 Travel: Widget Documentation
Author	Javier Rico

Approval Sign-off				
Owner	Role	Signature	Date	Version
Peter Offiong	Product Owner		08/11/21	2.6
Approver	Role	Signature	Date	Version
Avi Acharya	Leader of Product Development		08/11/21	2.6

Change History					
Doc Version	Widget Version	Status	Date	Author / Editor	Details of Change
1.4	3.0.14	Final	29 Oct. 19	Caitlin Inglis	Logo Update
1.5	3.0.15	Final	13 Nov. 19	Raghu Bhagawatula	Added Rescore and Rescreen Sections, Updated Widget Source section, modified Rescore URLs
1.6	3.0.15	Final	22 Jan. 20	Samuel Peach	Improvements made in line with other products
1.7	3.0.15	Final	25 Mar. 20	Samuel Peach	Improvements made following client queries.
1.8	3.0.15	Final	8 Apr20	Peter Offiong	Review and minor corrections – formatting and grammar
1.9	3.0.15	Final	22 Jul 20	Sam Peach	Added information regarding legacy -> RRT3 migration
2.0	3.0.15	Final	07 Oct 20	Sam Peach	Update URL versions and added RRT3 Flags information
2.1	3.0.15	Final	30 Nov 20	Carlos Sanchez	Improvements made following client queries.
2.2	3.0.16	Final	07 April 21	Carlos Sanchez	Updated to include encryption information
2.3	3.0.16	Final	30 April 21	Sam Peach	Rephrasing of encryption information
2.4	3.0.16	Final	11 August 21	Peter Offiong	Updating travel-specific features, exclusions and single or 2 character search

2.5	3.0.17	Final	01/11/2021	Javier Rico	New parameters, trigger groups, FAQ, changelog.
2.6	3.0.17	Final	08/11/2021	Peter Offiong	Updating BB3 to RRT3 & Blackbox to Risk Rating Tool
2.7	3.0.17	Final	16/02/2022	Francisco Figueiredo	Add Backup Declaration Add Conditions Unique Hash Code Add Dual Language Output
2.8	3.0.18	Draft	25/03/2022	Peter Offiong	Removal of trigger groups and document review
2.9	3.0.18	Final	29/03/2022	Francisco Figueiredo	Added new recommended integration option Added Save/Resume partial screening feature Adjustment to Multiple Language Output feature Added Migration Tool section Corrected minimum request parameters for registration Added sections to FAQ Added changelog
3.0	3.0.18	Final	22/09/2022	Francisco Figueiredo	Adjustment to Age parameter Adjustment to Gender parameter
3.1	3.0.18	Final	26/09/2022	Francisco Figueiredo	Rebrand
3.2	3.0.18	Final	24/10/2022	Estefania Giordano	Update Resume partial screening feature
3.3	3.0.18	Final	11/01/2023	Mikel Salazar	Added JSON output
3.4	3.0.19	Final	10/04/2023	Ulises Fernández	Updated document to reflect WCAG related improvements
3.5	3.0.19	Final	24/05/2023	Zhengyang Liu	Added Widget version tracking in Change History Updated code example

The latest approved version of this document supersedes all other versions, upon receipt of the latest approved version all other versions should be destroyed, unless specifically stated that previous version (s) are to remain extant. If any doubt, please contact the document Author.

1. Table of Contents

1	Introduction	7
	Purpose of the Risk Rating Tool Widget.....	7
	Screening Process Overview	7
1.1.1	Condition Search	7
1.1.2	Condition Screening.....	8
1.1.3	Declared Conditions Management	9
2	Prerequisites for integration	10
	Whitelisting	10
	Front-End Variables.....	12
	Server-Side Variables.....	12
3	Overview of how the RRT3 Widget works	12
	Placement.....	14
	Configuration	14
	Execution.....	15
	Interacting with the Widget.....	15
3.1.1	Integration based on JavaScript	16
3.1.2	Integration based on Forms.....	18
	Full example (JavaScript).....	18
	Authentication	22
3.1.3	Exchanging credentials for a temporary token.....	22
3.1.4	Providing the tokens to the widget.....	23
3.1.5	Example.....	24
4	Screening parameters	25
	Screening Parameters	25
	Example Register Screening Ticket.....	29
	Resume Screening	29
	Screening Data	30
	Examples.....	30
	Cancellation Risk Score Inputs.....	31
4.1.1	Lead Time Modifiers	31
4.1.2	Cost Modifiers.....	31

4.1.3	Example Register Screening Ticket with CRS	32
	RRT3 Travel-Specific Features	32
4.1.4	Annual Exclusion Feature	32
4.1.5	Winter Sports Exclusion Feature	33
4.1.6	Coronavirus (COVID-19)	34
	RRT3 Travel Exclusion Flags	34
4.1.7	Where are the exclusion flags raised?	35
4.1.8	Exclusion Flags	35
5	Results	36
	Risk Rating Tool Messages	36
	Screening Results	36
5.1.1	Travel XML Results	36
5.1.2	Cancellation XML Results	39
5.1.3	Travel JSON Results	42
5.1.4	Cancellation JSON Results	43
	Result Output Description	45
	Screening Settings	46
	Condition	47
	Multiple Language Output	48
6	API	49
	Introduction	49
	Authentication	50
	Recalculate/ Rescore	50
	Rescreen	52
7	Customization	55
8	Encryption	57
	General information	57
8.1.1	Purpose	57
8.1.2	Overview	57
8.1.3	Implementing decryption	57
8.1.4	Using a decryption endpoint	58
	Technical Details	59
8.1.5	Encryption and authentication	59
8.1.6	Key Derivation	59

8.1.7	Structure of the message.....	59
8.1.8	Decryption process	60
8.1.9	Decryption endpoint.....	61
	Rescreen and rescore.....	62
9	Save/Resume Screening	62
	General information.....	62
9.1.1	Save Screening	62
9.1.2	Resume Screening	64
	Technical Details.....	65
9.1.3	Save Screening	65
9.1.4	Resume Screening	68
10	Migration.....	70
10.1.1	Register	71
10.1.2	Rescore	72
11	Frequently Asked Questions	74
12	Samples	76
	Decryption.....	76
12.1.1	.NET	76
12.1.2	Python	79
12.1.3	Node.js	80
13	Versioning	82
14	Supported Browsers.....	83
15	Changelog.....	84
	Version 3.0.19.....	84
	Version 3.0.18.....	84
	Version 3.0.17.....	84
	Version 3.0.16.....	85

1 Introduction

Purpose of the Risk Rating Tool Widget

VRR's Risk Rating Tool 3 (RRT3) risk rating solution is implemented with two major components, the RRT3 web API and the RRT3 Widget. RRT3 encapsulates the data, business rules and calculation algorithms required to provide the medical risk score for each screening. The RRT3 Widget exposes a simpler point of integration for clients and implements an out of the box screening web user interface.

Screening Process Overview

The VRR screening UI consists of 3 main functional blocks that implement searching for conditions, screening a declared condition and managing the list of declared conditions.

This section provides an overview of these UI blocks taking the VRR Widget as an example.

1.1.1 Condition Search

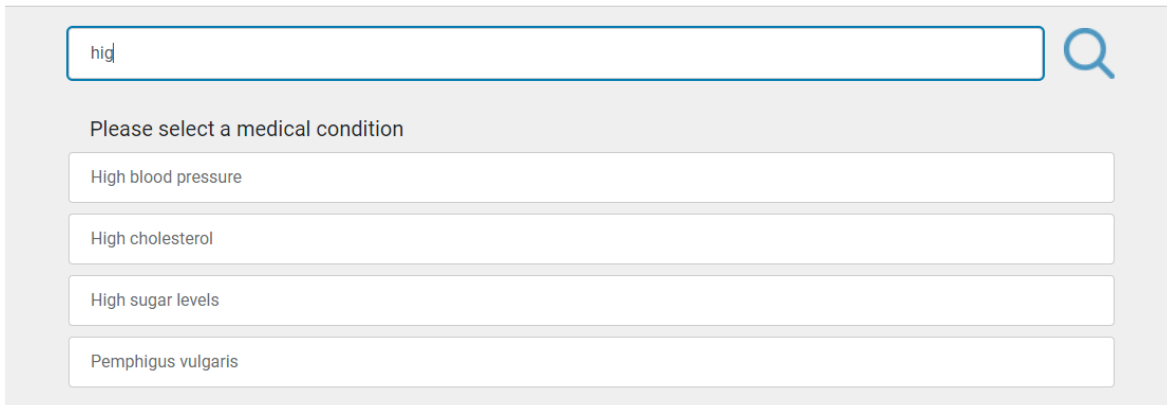
RRT3 provides functionality to search condition names and their synonyms. The Widget implements this using a search textbox and a result list that is updated as the user enters new characters into the search text box. It is important to note that the result list will be displayed typically after a minimum of 3 characters for a condition has been keyed into the search textbox. However, there are special cases where either a single character or 2 characters can return the result list.

RRT3 employs the following logic for condition search:

1. A 1-character search may be required for some non-English languages and this either returns a 100% match or an empty result list
2. A 2-character search occurs by looking for a recognised medical abbreviation (such as AF for Atrial Fibrillation) that returns a 100% match for the 2 characters inputted by the user, else an empty result list is returned
3. A 3-character search will return a partial match, which will include a result list that contains the specified 3 letters

The end-user will declare the condition into the Declare New Condition method.

Search



The screenshot shows a search interface with a text input field containing 'hig' and a magnifying glass icon to its right. Below the input field, the text 'Please select a medical condition' is displayed. Underneath this text, there are four rectangular buttons, each containing a medical condition: 'High blood pressure', 'High cholesterol', 'High sugar levels', and 'Pemphigus vulgaris'.

Figure 1: 3-character search

The matching search term must:

- Be a condition or synonym.
- Be allowed to show (as per class overrides configuration) and not hidden.

Screen reader users will hear a text that prompts them to enter the term to be searched for. As they type, they will hear the entered character followed by the whole search term and the number of results produced. If the user needs to use the keyboard, pressing the tab key will lead them to the search button, and pressing it twice or more times will allow them to select one of the results available. This process can be repeated if needed to declare more conditions.

Work has also been done in the visual interface for improving the contrast of focused areas, having users who suffer visual disabilities in mind.

1.1.2 Condition Screening

Users complete a declaration by answering a set of questions that appear in sequential order.

Only one question can be active (i.e. be in edit mode) at a time. It is possible to edit previously answered questions and most questions ask for a single selection from multiple options. Nonetheless, some questions allow for multiple answers to be selected.

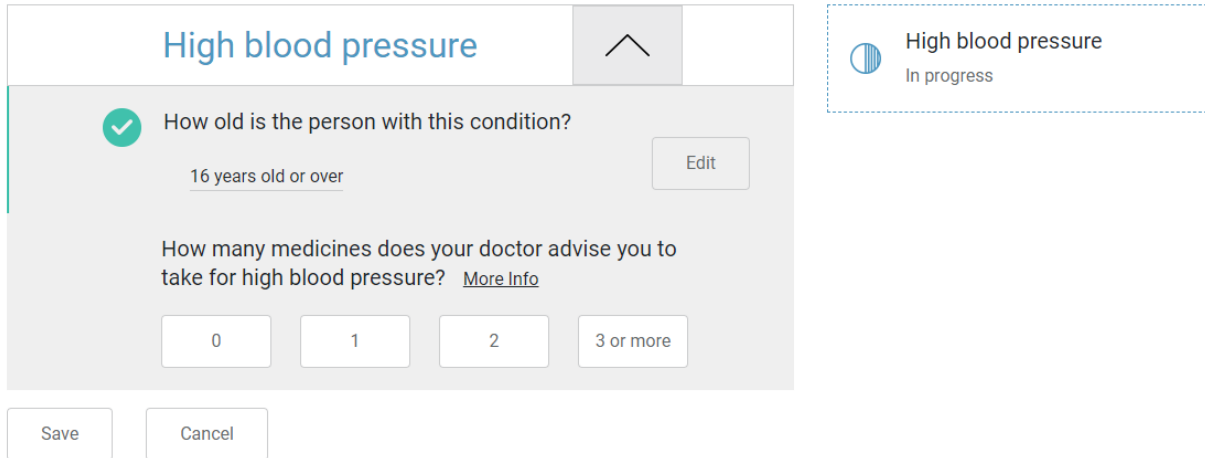


Figure 2: Edit mode

Some answers, within the database, result in a 'Forward' that leads the user to another condition. The declaration is complete only when all questions in all forwarded conditions have been answered.

There is also the possibility to save a screening at any stage for a later resume. This can be done at any point of the Q&A. This is an optional feature that can be enabled at a user level. See section 9. for additional information.

Please note, some conditions might not have any questions associated with them and the following message will be returned for such conditions 'There are no questions for this condition. Please press continue and complete your medical declaration'. Customer facing documentation should be carefully considered for conditions where no questions are asked.

If a screen reader is used, it will read the questions aloud and tell the users which options are grouped and checked or selected. The user can navigate through the options using the arrow keys. For questions that allow multiple answers, the user can press enter or spacebar to select or deselect as many options as desired. For questions that only support one answer, spacebar or enter will choose the selected option.

In a list of conditions, users who rely on a screen reader can navigate through collapsed/expanded panels getting notified about the status of each panel.

1.1.3 Declared Conditions Management


This section allows the user to review the list of declared conditions, initiate editing, or the removal of conditions from the list. This is also the place where the user can select to finish the whole declaration process. The user can also save the screening in the current state for later resume before answering the questions for any declaration or after all declaration have the questions answered. This is an optional feature that can be enabled at a user level. See section 9. for additional information.

The 'Edit' and 'Remove' functionality applies to Condition Groups. As the name implies, Condition Groups can consist of one or more conditions. Conditions are grouped together as one when a medical condition forwards to another condition e.g., Mini stroke forwarding to Cerebrovascular accident as seen in the image

below. This linkage implies that if a condition within a group is excluded, all conditions within that condition group will be excluded from medical cover. A group that manages the “Remove” functionality can consist of one or more groups for editing.


Once a declaration has been completed, it is possible to return and amend where required; this process is called rescreening. This is different from the rescore function, which allows the user to submit their declaration for scoring again, without entering the Risk Rating Tool to review or amend the previously declared conditions; the rescore function can also be used to change the declaration’s input parameters. During a rescore, if there have been changes to the Risk Rating Tool content since the original screening was made, conditions may be forced to be edited or screened anew. In this case, the list of ‘Declared Conditions’ will be annotated with warning messages and these ‘Declared Conditions’ will require rescreening (editing or removing) to complete the rescreen and submit for scoring. The ‘Declared Conditions’ section handles this when required.

Search



List of conditions

Declared conditions

High blood pressure	Edit	Remove
Mini stroke	Edit	Remove
 Cerebrovascular accident		

If you would like to exit, please press Complete Declaration.

Save Declaration

Complete Declaration

Figure 3: Declared conditions management

2 Prerequisites for integration

Whitelisting

For all widget integrations, due to Same-Origin Policy and CORS, Verisk will need to whitelist the client’s domain names. This process needs to be done for both UAT and production environments. As per Verisk corporate security policies, insecure traffic is not allowed into gateway APIs. Therefore, we can only whitelist legitimate, secure websites having a DNS (internal or external) entry; also, development on the local host will work without whitelisting (e.g. `https://localhost/BlackBox3`). All the websites should be hosted as ‘HTTPS’ and hence will use ‘port 443’; no other port is allowed. Please allow 4 working days for the whitelisting process.

Verisk's technical team will provide the different variables once the client's technical team is ready to commence integration with the RRT3 Widget; these are listed in 2.2 and 2.3 below.

To begin a UAT integration Verisk will require:

- UAT domain names – The client's technical team will need to provide domain names where the UAT integration will be hosted; these will need to be whitelisted as above before the build can commence.

Verisk will then provide the following for a UAT integration:

- RRT3 Username and Password – The client will be initially provided with a username with the suffix 'UAT' to commence their UAT integration. Please note that any RRT3 logins ending in 'Demo' are for demonstration only and should not be used for integration.
- UAT Licence Key – The login's licence key to access the Verisk servers.
- Gateway User ID – Another authentication process to access the RRT3 API.
- URL – To connect to the RRT3 API.
- Sample.zip Folder – This folder contains the HTML placeholder file.
- List of Region IDs specific to your RRT3 Travel integration and location.
- Decryption passphrase or licence key

Post-audit, the client can promote to live and will need to provide production domain names. Verisk will provide the following:

- Production RRT3 login and password.
- Production Licence Key.
- Gateway User ID and password.
- Verisk will also whitelist the client's production domain names.

Important: No test screenings, including load balancing tests, should be completed on the Production server unless agreed by Verisk. Please contact your Client Services team for more information if required. Additionally, we will require production domains and sub-domains to be shared in advance to ensure this can be whitelisted according to our SLAs.

Front-End Variables

Variable	Description
VERISK_GATEWAY_API	The URL used by the Widget to connect to the RRT3 API
BB3_WIDGET	The example HTML placeholder files, provided by Verisk, that will load the RRT3 Conditions, Questions and Answers
VERISK_WIDGET_USER	The username for the RRT3 API, provided by Verisk

Server-Side Variables

Variable	Description
VERISK_GATEWAY_USERID	A user string, provided by Verisk, that is used to retrieve the token required by RRT3 API for authentication
VERISK_GATEWAY_PASSWORD	A password string, provided by Verisk, that is used to retrieve the token required by RRT3 API for authentication
VERISK_LICENSE_KEY	A licence key string, provided by Verisk, that is used by the widget to authorise itself with RRT3 APIs
VERISK_INTERNAL_LICENSE_KEY	An authentication key string, provided by Verisk, invoked for specific internal Verisk endpoints. Should be kept a secret and should never be shared with end-users. At the time of this writing, this authentication key is only used to call the <code>/decrypt</code> endpoint or allows the client to decrypt the content themselves. See section 8 for a detailed description.

Please note that these Gateway Credentials will get locked after 3 incorrect attempts.

3 Overview of how the RRT3 Widget works

The Widget provides the functionality to perform medical screening for a single user by collecting answers to questions related to any number of pre-existing conditions declared by the customer.

When your application needs to perform a screening, it should render the Widget by providing:

- The necessary details (such as Age and medical Q&A history, if this is rescreening a RRT3 result)

- A 'POST' URL where the result of screening will be posted.
- A 'POST' URL for getting a temporary Verisk token.

Your application will be provided Risk Rating Tool credentials for generating temporary Verisk tokens. These credentials should be protected and not provided to the widget directly. This implies that your application should provide an authentication mechanism outside of the widget. The widget is configured to call the application's authentication service, thus providing the functionality to call for a temporary token and handle expired tokens as needed.

Upon rendering, the RRT3 Widget will register a request for screening from your application and will return a screening identifier that can be tracked throughout the screening session. There is also an optional feature to resume a previously saved screening, depending on the parameters passed to the RRT3 Widget.

Once registered for screening, the customer arrives at the Conditions page. RRT3 provides the functionality to search condition names and their synonyms. The widget implements this using a search textbox and a result list that is updated as the user enters new characters into the search text box. They can select the condition and either proceed to declare another condition, begin the declaration for the condition or save the current state of the screening by clicking the 'Save Declaration' button. Upon clicking on the 'Start declaration' button, the customer is shown a set of questions that appear in sequential order for the selected condition and they can select the appropriate answer/s for these questions – which is/ are also shown by the Risk Rating Tool. These set of questions are also able to be saved by clicking the 'Save' button for later resume as an optional feature.

In case the user passed arguments that are eligible for a resume screening, a call to the RRT3 API will be done and the RRT3 Widget will restore the screening in the state where it was previously saved in case the saved screening is not expired.

The RRT3.0 widget uses VRR's RRT3.0 APIs to search for conditions and to build the necessary sequence of questions from the VRR medical screening knowledge base. No personal details of the customer are to be transferred to the RRT3.0 widget.

After the customer has finished the declaration for all conditions, the customer needs to click on the 'Complete Declaration' button. When this is done, the widget calls the RRT3.0 API, which in turn calculates the risk scores using its algorithm and 'POSTs' the completed screening result either in XML or JSON format on the Post action specified in the form where the widget is hosted.

The output with Screening results contains the below-listed scores:

- Medical Risk Score
- Cancellation Risk Score – depending on VRR Agreement and Terms.

For a description of what the above scores mean and how can they be used, please refer to the VRR Guide to Implementation of Travel Risk Rating Tool.

In addition to the above scores, the output also consists of:

- ICD codes (ICD-9 and ICD-10),
- The full list of questions and answers as completed by the customer

- The customer journey details (start and end time of declaration for each condition) within the Risk Rating Tool could be saved in the client's application and used if needed to rescreen the customer at a future time

Placement

The widget requires our own JavaScript plugins as well as a stylesheet to work properly.

Add the stylesheet link into your head to load our CSS. This is required even if you want to customize the styles (please refer to the Customization section for that).

```
<link rel="stylesheet" href="https://blackbox30-travel.verisk.com.au/Content/BootstrapVerisk-3.0.18.css" />
```

Place the following `<script>` tag near the end of your pages, after the container and the configuration variables are defined on the page, to enable the widget.

```
<script src="https://blackbox30-travel.verisk.com.au/scripts/blackbox-3.0.18.js"></script>
```

Finally, on the place where the widget will be rendered, define the following HTML element:

```
<div class="verisk-container" id="BlackBoxContainer"></div>
```

Configuration

The widget initialization and behaviour is controlled by the following variables that need to be provided on the global scope:

Mandatory Variables:

API_VERSION – Version of the API to use. Should be V2

ROOT_URL – RRT3 Web API endpoint

HOSTPAGE_ROOT_URL – Relative path to the page where the widget is being added.

EVENT_LISTENERS / RESULT_DESTINATION_VAR – As explained in section 3.4, the event system takes precedent over the form submission, but one of these variables should be provided to retrieve the results. The `EVENT_LISTENERS` should have a function `onFinish`. The `RESULT_DESTINATION_VAR` should contain the ID of the input element used for collecting screening results

USER_SETTINGS – JSON objects containing all settings required to initiate the screening session (see Section 4.0 - Screening Parameters)

AUTH_DATA – authorization parameters, for example:

```
var AUTH_DATA = {
  //url of the application endpoint to get temporary gateway token
  uri: 'https://<YOUR ENDPOINT HERE>',
  // This custom data will be sent to your endpoint (as a body) when re
  trieving credentials.
```

```
// This is optional, and can be left null or empty if not needed.
//e.g.data: null or data: {}
data: {
    //grant_type: "password"
},
config: {
    // These custom headers will be sent to your endpoint when retrieving credentials.
    // This is optional, can be left null or empty if not needed.
    headers: {
        authorization: "<DEPENDING ON YOUR ENDPOINT SECURITY>"
    }
}
};
```

Optional Variables:

PROGRESS_DESTINATION_VAR – ID of the input element used for collecting intermediate screening state

A special note about `AUTH_DATA`. This data is related to your own authentication endpoint (see section 3.6 for more details). It simply controls the requests that the widget will perform to your endpoint to get credentials. The URL, as well as the data (body) and config (headers), needs to be configured based on how you have set up that endpoint. For example, if your endpoint uses a Bearer token to identify the user, you should use an `Authorization` header with the `Bearer` token.

Execution

For traditional web applications (or multi-page applications, MPA), simply by including our `<script>` tag and providing the configuration is enough to get started.

But for single-page applications (SPA) this process is not enough, as the widget should be re-rendered without page reload. In this case, the widget can be accessed as a JavaScript variable `bb3Widget` with an exposed parameterless function `run()`. This function can be called each time the Widget container is rendered in the client's SPA.



Interacting with the Widget

The basic interaction with the Widget will involve your system retrieving the calculation results at the end of the screening process. You can do this in two ways: a JavaScript-based approach, where you provide some functions that will be called at the end of the screening process; or a form-based approach.

For modern applications, Verisk recommends the JS-based approach, which provides additional benefits and integration opportunities. If both are used, the JS-based approach will take priority.

3.1.1 Integration based on JavaScript

As described in the configuration section, you can provide an `EVENT_LISTENERS` global variable containing an object with up to two properties specifying functions. The widget will invoke these functions at specific moments during the customer journey, allowing for communication between the widget and your application.

`onFinish`: this function will be invoked when the screening process has finished. It receives two parameters: `eventType`, a string which details the event that triggered the finishing of the screening, and `payload`, which depending on the event type will contain different information.

There are two available events for `onFinish`:

1. `COMPLETE_SCREENING`: Invoked when a screening has been completed. The payload will contain the screening result.
2. `SAVE_SCREENING`: Invoked when a screening has been saved. The payload will contain the partial completed screening.

`onError`: this function will be invoked when an unrecoverable error has occurred. It receives two parameters: `eventType`, a string which details the event that triggered the finishing of the screening, and `errors`, an object described below.

There are two available event types for `onError`:

1. `SAVE_SCREENING`: Invoked when an error occurs while saving a screening.
2. `RESUME_SCREENING`: Invoked when an error occurs while resuming a screening.

The `errors` variable follows the JSON:API specification. Whenever an unrecoverable error(s) occurs after a request to RRT3 when triggering the above-mentioned events, the Widget provides its own error model based on the responses from RRT3. All errors have a `code` property to identify the error code and a `detail` property to describe the error.

The error model can be checked hereunder:

EVENT TYPE	WIDGET ERROR CODE	DESCRIPTION
SAVE_SCREENING	SAVE_SCREENING_INVALID	When a request is invalid and RRT3 is not able to process the request.

RESUME_SCREENING	RESUME_SCREENING_INVALID	
RESUME_SCREENING	RESUME_SCREENING_EXPIRED	When the provided saved screening is already expired. This will display a message to the user with a modal dialog
SAVE_SCREENING	SAVE_SCREENING_UNEXPECTED	When an unexpected error is returned from RRT3
RESUME_SCREENING	RESUME_SCREENING_UNEXPECTED	

Below is an example of the `errors` array passed as an argument to the `onError` function:

```
[
  {
    code: "RESUME_SCREENING_EXPIRED",
    detail: "Resume data is expired"
  }
]
```

Verisk may add new event types in the future as we expand the functionality for the product. While any new functionality will be released on a new version, we recommend that your code is prepared to deal with unexpected event types.

The `COMPLETE_SCREENING` is currently only used on the `onFinish` event and will finish the interaction with the Widget giving the control back to the client to take any action deemed necessary.

The `RESUME_SCREENING` is currently only used on the `onError` event and will finish the interaction with the Widget giving the control back to the client to take any action deemed necessary.

The `SAVE_SCREENING` is currently used on the `onFinish` and `onError` events. The `onFinish` event will finish the interaction with the Widget giving the control back to the client to take any action deemed necessary. The `onError` event will allow the capture of the errors but will allow the user to do further actions on the Widget.

An example of using this functionality

```
var EVENT_LISTENERS = {
  onFinish: function(eventType, payload) {
    switch (eventType) {
      case 'COMPLETE_SCREENING':
        // The payload variable will contain the screening result.
        break;
      case 'SAVE_SCREENING':
        // The payload variable will contain the partial screening result.
        break;
    }
  }
}
```

```
        default:
            // Your code should allow for unexpected or new event types.
            break;
    }
},
onError: function(eventType, errors) {
    switch (eventType) {
        case 'SAVE_SCREENING':
        case 'RESUME_SCREENING':
        default:
            // All the errors will follow the format described above.
            // You can differentiate between them based on your needs.
            break;
    }
    return errors;
}
};
```

3.1.2 Integration based on Forms

If the form submission option is used the variable `EVENT_LISTENERS` can't be provided and the widget needs to be rendered within a form element, which also contains the input control used for collecting the screening results.

For example, Widget Collecting Result

```
<form action="/Screening/FinishScreening/" method="post" role="form">
  <input id="scrResult" name="scrResult" value="" type="hidden">
</form>
```

When the user completes the screening process, the widget will send a request to the RRT3 API, and a response containing the result of the declaration (including the result) will be returned. The Widget will then place the result of the request into an element inside the form. Which form element it uses will depend in the configuration variable `RESULT_DESTINATION_VAR` (in the above example, it would be element `#scrResult`).

After the result is placed into the element, the form will be submitted. By default, this form submit process will result in a POST call being made to whatever action element the form has (in the above example, `/Screening/FinishScreening`). You can provide your own server endpoint to catch this result. But it is also possible to catch the event and not submit the form, which is the recommended flow for SPAs.

Full example (JavaScript)

```
<!doctype html>
<html lang="en">
```

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="Description" content="Verisk Risk Rating - Widget Demo">

  <link rel="stylesheet" href="https://blackbox30-
traveluat.verisk.com.au/Content/BootstrapVerisk-3.0.18.css">
  <title>Verisk Risk Rating - Widget Demo</title>
</head>

<body>
  <form class="widget-form" method="post" role="form">
    <div class="verisk-container" id="BlackBoxContainer"></div>
  </form>

  <script type="text/javascript">

    var API_VERSION = 'v2';
    var ROOT_URL = 'https://gatewayuat.verisk.com/rating/au/travel/';
    var HOSTPAGE_ROOT_URL = window.location.pathname;
    var USER_SETTINGS = {
      $G_Age: 30,
      $G_TripDuration: 8,
      $G_LeadTime: 23,
      $G_IsWinterSport: 1,
      $G_IsAnnual: 1,
      $G_DirectlyLinked: false,
      $G_IndirectlyLinked: false,
      $G_Regions: "1",
      $G_CancellationCostId: 4,
      $G_Username: "<CLIENT USERNAME>"
    };

    // Below information should be provided application
    // Also expected response needs to be as below:
    // {"access_token":"XXX","client_jwt":"XXXX"}
    // where access_token: gateway token and client_jwt: provided license key
    var AUTH_DATA = {
      //url of the application endpoint to get temporary gateway token
      uri: 'https://<YOUR ENDPOINT HERE>',
      // This custom data will be sent to your endpoint (as a body) when retrieving cred
entials.

      // This is optional, and can be left null or empty if not needed.
      //e.g.data: null or data: {}
    };
  </script>
</body>
```

```
data: {
    //grant_type: "password"
},
config: {
    // These custom headers will be sent to your endpoint when retrieving credentials.

    // This is optional, can be left null or empty if not needed.
    headers: {
        authorization: "<DEPENDING ON YOUR ENDPOINT SECURITY>"
    }
}
};

// Client events to be configured by the client
// If the client doesn't set any event there should be a RESULT_DESTINATION_VAR configured to get the results.
// For the Resume or Save partial screenings feature these events must be used to retrieve the Save result and to return the control to the client
// There are 2 possible events which receive a type depending on the workflow and user action
// onFinish event is triggered when the user finishes the interaction with the widget, such as Complete Screening or Save Screenings
// onError event is triggered when Resuming or Saving screenings if an error occurs during these processes
var EVENT_LISTENERS = {
    onFinish: function(eventType, result) {
        switch (eventType) {
            case 'COMPLETE_SCREENING':
                console.log(`Complete screening: ${result}`);
                break;
            case 'SAVE_SCREENING':
                console.log(`Save screening: ${result}`);
                break;
            default:
                console.log(result);
                break;
        }
        return result;
    },
    onError: function(eventType, errors) {
        switch (eventType) {
            case 'SAVE_SCREENING':
                for (var error of errors) {
                    switch (error.code) {
                        case 'SAVE_SCREENING_INVALID':
```

```
        console.log(`Save screening invalid: ${error.detail}`);
        break;
    case 'SAVE_SCREENING_UNEXPECTED':
        console.log(`Save screening unexpected: ${error.detail}`);
        break;
    default:
        console.log(`Save screening: ${error.detail}`);
        break;
    }
}
break;
case 'RESUME_SCREENING':
    for (var error of errors) {
        switch (error.code) {
            case 'RESUME_SCREENING_INVALID':
                console.log(`Resume screening invalid: ${error.detail}`);
                break;
            case 'RESUME_SCREENING_UNEXPECTED':
                console.log(`Resume screening unexpected:
${error.detail}`);

                break;
            case 'RESUME_SCREENING_EXPIRED':
                console.log(`Resume screening expired: ${error.detail}`);
                break;
            default:
                console.log(`Save screening: ${error.detail}`);
                break;
        }
    }
    break;
default:
    console.log(`Default: Code: ${errors[0].code}; Message:
${errors[0].detail}`);
    break;
}
return errors;
}
};
</script>
<script src="https://blackbox30-traveluat.verisk.com.au/Scripts/blackbox-
3.0.18.js"></script>

</body>
</html>
```

Authentication

The Widget communicates with the RRT3 API to operate. Those requests require two tokens to be present: a temporary token, to be renewed hourly; and a permanent token. The temporary token can be exchanged by using the Verisk-provided credentials. However, because the widget is a client application, if those credentials were to be directly provided, they could be easily stolen, and a malicious user could then make requests on your behalf. That is why the widget requires a specific endpoint to be implemented by your server-side system, so it has the credentials safely stored and only provides temporary tokens.

This endpoint will need to exchange Verisk-credentials for a temporary token, and then returning that token as well as the permanent licence key. You should restrict access to this endpoint so that only an authenticated and authorized user, according to your application requirements, can invoke it.

3.1.3 Exchanging credentials for a temporary token

To retrieve the temporary authentication token, your system will need to perform a request using the credentials provided by VRR:

```
POST https://gatewayuat.verisk.com/token
```

Body

```
grant_type=client_credentials
```

Headers

```
Authorization      Basic <GATEWAY_CREDENTIALS>
```

The "Basic" HTTP authentication scheme is defined in RFC 7617, which transmits credentials as user ID/password pairs, encoded using base64.

For example, USER:PASS would be encoded using base64 becoming VVNFUjpQQVNT

```
Content-Type       application/x-www-form-urlencoded
```

```
Accept             application/json
```

A sample token-retrieval request:

```
curl 'https://gatewayuat.verisk.com/token' \
-X POST \
-H 'Authorization: Basic <GATEWAY_CREDENTIALS>' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'Accept: application/json' \
```

```
-d 'grant_type=client_credentials'
```

The above command returns JSON structured like this:

```
{
  "scope": "default",
  "token_type": "bearer",
  "expires_in": 3600,
  "access_token": "33acbd1f0f1d46249f5b90bf7c63ad3d"
}
```

The temporary authentication token is contained in the `access_token` field, and the time for expiration (in seconds) is contained in the `expires_in` field.

Further details can be found in <https://gateway.verisk.com/docs/Authentication.ashx>

REMEMBER: The gateway credentials must be provided as User and Pass joined by the `:` character, and then encoded in base64. Verisk will provide you with the following information:

Credential	Description
Gateway API username	Username for the Gateway API (for requesting temporary bearer tokens)
Gateway API password	Password for the Gateway API (for requesting temporary bearer tokens)
RRT3 API Licence Key	Token for RRT3 API
RRT3 User Login	Login id created by Verisk specific to RRT3 Travel. Should be provided inside the requests, as the <code>\$G_Username</code> parameter.

3.1.4 Providing the tokens to the widget

The widget will perform a POST request to your endpoint (specified in the URI field of the AUTH_DATA variable). It expects your endpoint to give a response like:

```
{
  "access_token": "XXXXXXXXXXXXXXXXXX",
  "client_jwt": "YYYYYYYYYYYYYY"
}
```

Where `access_token` is the temporary access token obtained as previously described, and `client_jwt` is your licence key.

The widget can optionally send additional data with the request, such as the request body or headers. You should use this functionality to implement authentication or anything else that your server system may require. See section 3.4 for further details.

3.1.5 Example

Below is a sample implementation for the Authentication endpoint that the application will provide for the RRT3 widget to get a temporary token to use the RRT3 API Services (in C#):

```
[Authorize]
[HttpPost]
public ActionResult AuthToken(userdata data)
{
    //dummy authentication logic for client portal
    if (Validate(data))
    {
        var token = GetToken();
        if (token != null)
        {
            token.client_jwt= @VERISK_LICENSE_KEY
            return Json(token);
        }
    }

    return new HttpUnauthorizedResult("Invalid Credentials");
}

private TokenDetails GetToken()
{
    HttpClient client = new HttpClient();
    client.DefaultRequestHeaders.Authorization =
        new AuthenticationHeaderValue(
            "Basic",
            Convert.ToBase64String(Encoding.ASCII.GetBytes(
                @VERISK_GATEWAY_USER + ":" + @VERISK_GATEWAY_PASSWORD
            ))
        );
    client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

    HttpRequestMessage message = new HttpRequestMessage(HttpMethod.Post, @VERISK_GATEWAY_API);
    message.Content = new StringContent("grant_type:client_credentials", Encoding.UTF8, "application/x-www-form-urlencoded");
}
```



```
var tokenResult = client.SendAsync(message).Result;

if (tokenResult.IsSuccessStatusCode)
{
    var tokenResponse =
        JsonConvert.DeserializeObject<TokenDetails>(tokenResult.Content.ReadAsStringAsync().Result);
    return tokenResponse;
}

return null;
}

public class TokenDetails
{
    public string access_token { get; set; }
    public string client_jwt { get; set; }
}
```

4 Screening parameters

Settings that are required to initiate a screening session should be provided via the `USER_SETTINGS` variable.

Screening Parameters

Parameter Name	Type	Description	Required	Example Value
\$G_Username	String	Client's specific RRT3 Login	Mandatory	"ClientLogin" – see Section 2.2
\$G_IsAnnual	Integer	Defining policy type: 1 = Single Trip 2 = Annual Multi-Trip 3 = Both	Mandatory	1
\$G_IsWinterSport	Integer	Defining policy type: 1 = No	Mandatory	1

		2 = Yes 3 = Both		
\$G_TripDuration	Integer	An integer value for trip duration	Optional	"7"
\$G_LeadTime	Integer	An integer value for the lead time until trip; see Cancellation Risk Score Inputs below for values.	Optional	"1"
\$G_CancellationCostId	Integer	Integer value to define a cancellation cost bracket; see Cancellation Risk Score Inputs below for values	Optional	1
\$G_DirectlyLinked	Boolean	Whether to include directly linked conditions. Responses can be 'True' or 'False'	Mandatory	False
\$G_IndirectlyLinked	Boolean	Whether to include indirectly linked conditions. Responses can be 'True' or 'False'	Mandatory	False
\$G_Regions	String	One or more region IDs are separated by a comma.	Mandatory	"201, 02, 203, 204, 205"
\$G_ShowClasses	String	A string of Condition Class Ids. Control whether Pregnancy Conditions and/or Complications and/or Covid-19 are exposed or hidden Complications of pregnancy: ClassId = 3 Pregnancy Conditions: ClassId = 4 Covid-19: ClassId = 6	Optional	"3" (showing only "Complications of Pregnancy") "3,4" (Show both) "6" (Show Covid-19)
\$G_HideClasses	String	A string of Condition Class Ids. Control whether Pregnancy Conditions	Optional	"4"

		<p>and/or Complications and/or Covid-19 are exposed or hidden</p> <p>Complications of pregnancy: ClassId = 3</p> <p>Pregnancy Conditions: ClassId = 4</p> <p>Covid-19: ClassId = 6</p>		<p>(hiding "Pregnancy conditions")</p> <p>"3,4"</p> <p>(Hide both)</p> <p>"6"</p> <p>(Hide Covid-19)</p>
\$G_Age	Integer	Age in years at the time of coverage. Age parameter can be added to \$G_ResultDisplayParameters with value "Age" to be added to the result.	Optional	35
\$G_IsMetric	Boolean	Whether questions that ask weight and height are by default set to Metric instead of Imperial units.	Optional	true
\$G_RequestedExclusions	Space Separated String	<p>Configuration of Condition ICD codes that cause a medical conditions within the declaration to be excluded.</p> <p>When one of this Conditions is declared, the screening is allowed to finish. The declaration group is excluded with a score of 1.</p>	Optional	110
\$G_IsScreeningLite	Integer	This enables the risk rating lite question set and is enabled on login level.	Optional for RRT3 Mandatory for Legacy XMLs	0 or 1 (0 for false & 1 for true)
\$G_PartialScreening	String	Content from the response from the Save Screening optional feature.	Optional	

		<p>Parameter used with the \$G_Username parameter to restore a previously saved screening</p> <p>When initializing the Widget, it will search for this parameter and if it is set and has a value, the widget will try to resume a screening by making a request to the RRT3 API and will try to restore the saved screening to the state where it was initially saved.</p> <p>Check section 9.1.2. for a more detailed explanation</p>		
\$G_Gender	String	<p>Gender, Male or Female. Gender parameter can be added to \$G_ResultDisplayParameters with value "Gender" to be added to the result.</p>	Optional	"Male"
\$G_ResultDisplayParameters	String Array	<p>An array of strings with the parameters that should be returned in the result.</p> <p>The possible values are:</p> <ul style="list-style-type: none"> • Age • Gender 	Optional	["Age", "Gender"]
\$G_ResultFormat	String	<p>The desired format for the screening result. It can be "xml" or "json".</p>	Optional. If not provided, the result will be returned using the format specified in the account's configuration	"\$G_ResultFormat": "json"

			on. If both formats are specified in the account's configuration, XML will take precedence.	
--	--	--	---	--

If an optional parameter is not being used, then the parameter should not be included in the request.

Example Register Screening Ticket

```
{
  "$G_Age": 19,
  "$G_DirectlyLinked": true,
  "$G_IndirectlyLinked": false,
  "$G_Regions": "201",
  "$G_IsAnnual": 1,
  "$G_IsWinterSport": 1,
  "$G_Username": "ClientLogin"
}
```

Resume Screening

```
{
  "$G_PartialScreening": "<BASE64 ENCODED STRING>",
  "$G_Username": "<YOUR USERNAME>",
  "$G_Parameters": {
    "$G_IsAnnual": 2
  }
}
```

To initiate a Resume Screening, the `$G_PartialScreening` and `$G_Username` parameters must be specified. The `$G_PartialScreening` parameter must be populated with the result from a previously saved screening.

If user has Include Policy Information feature disabled, new parameters must be sent in the `$G_Parameters` object. This should include at least the mandatory ones, but they can also add optional parameters.

For a user that has Include Policy Information feature enabled, they can override the parameters by providing the specified ones to override, override all or just keep the previous information.

Screening Data

To perform a rescore or a rescreen, the `$G_ScreeningData` parameter must be specified and populated with a previous screening. This will be either the content from the `<SystemData>` node (which is a base64 string) or the content from the `<ScreeningPath>` node (which needs to be encoded as base64 before passing it as `$G_ScreeningData`). RRT3 will recognize this as a previous declaration and use that as input.

If encryption is activated, you can pass the entire encrypted result as the input for `$G_ScreeningData`. In this case, there is no need to perform any additional encoding or decoding, you can just get the relevant string from the calculate response and pass it as-is. RRT3 will decrypt the content using the proper credentials, and then handle it as needed.

During rescreen, because of possible changes in the Risk Rating Tool content since the original screening have been made, conditions may be forced to be edited or screened from scratch. In this case, the list of Declared Conditions will be annotated with warning messages and these Declared Conditions will require editing or rescreening to complete the Rescreen and submit for Scoring. The Declared Conditions section handles this when required.

Examples

Minimal registration, using the Wintersports, Single Trip policy type and the Domestic/Low Risk region:

```
USER_SETTINGS = '{"$G_IsWinterSport": 1, "$G_IsAnnual": 1, "$G_TripDuration": 39, "$G_LeadTime": 1, "$G_Regions": "201, 301", "$G_Username": "ClientLogin", "$G_DirectlyLinked": "true" }'.
```

As above, but also including directly linked conditions

```
USER_SETTINGS = '{"$G_IsWinterSport": 1, "$G_IsAnnual": 1, "$G_TripDuration": 39, "$G_LeadTime": 1, "$G_Regions": "201, 301", "$G_Username": "ClientLogin", "$G_DirectlyLinked": "true" }'.
```

Initiating rescreening for a previous result:

```
USER_SETTINGS = '{"$G_IsWinterSport": 1, "$G_IsAnnual": 1, "$G_TripDuration": 39, "$G_LeadTime": 1, "$G_Regions": "<Value>", "$G_Username": "ClientLogin", "$G_ScreeningData": "H4sIAAAAAAEAK1UzU/bMBT/V9586SVB+
```

to7mzO4uiVs3fb+r1M/dyaOPkLaEdviW0GwkmT93ppv3Es2kvfJhNv51dzibTAgYvJHwAJQrFjde
JSh8AKUjCg98GAAA="}' .

Cancellation Risk Score Inputs

Cancellation Risk Score is an optional feature in RRT3 Travel, and it is the output of the cancellation risk rating exercise. To receive a Cancellation Risk Score, the following inputs must be provided:

4.1.1 Lead Time Modifiers

Lead Time is the number of days left to the trip start date (Trip start date – Current date). The following multiplier values should be sent in depending on the lead time.

N.B. Please confirm the lead time value thresholds to apply for each LTM with your business unit.

Lead Time Modifier IDs	Description
0	Lead Time Modifier 0
1	Lead Time Modifier 1
2	Lead Time Modifier 2
3	Lead Time Modifier 3
4	Lead Time Modifier 4
5	Lead Time Modifier 5
6	Lead Time Modifier 6

LTM 0 is intended for use when the departure date is immediate and there is no risk of cancellation for the customer's trip. LTM 0 will always return a CRS = 1.

4.1.2 Cost Modifiers

The Cost Modifier value should be used in line with the cancellation benefit available to the specific customer, the higher the cancellation benefit, the higher the Cost Modifier.

N.B. Please confirm the cancellation benefit value thresholds to apply for each Cost Modifier with your business unit

Cost Multiplier IDs	Description
1	Cost Modifier 1
2	Cost Modifier 2
3	Cost Modifier 3
4	Cost Modifier 4
5	Cost Modifier 5
6	Cost Modifier 6

4.1.3 Example Register Screening Ticket with CRS

```
{
  "$G_Age": 19,
  "$G_DirectlyLinked": true,
  "$G_IndirectlyLinked": false,
  "$G_Regions": "<Value>",
  "$G_IsAnnual": 1,
  "$G_IsWinterSport": 1,
  "$G_LeadTime": 1,
  "$G_TripDuration": 39,
  "$G_CancellationCostId": 1,
  "$G_Username": "ClientLogin"
}
```

RRT3 Travel-Specific Features

4.1.4 Annual Exclusion Feature

VRR has identified several conditions that are not suitable for cover on an annual multi-trip policy and the Annual Exclusion Feature (AEF) allows VRR's clients to exclude these from cover. However, VRR recognises that not all clients want to use this feature; please find specific instructions on how to use or not use this feature below.

Parameter: "\$G_IsAnnual"

Type: Integer

Status: Mandatory parameter required for all screenings

This parameter relates solely to the application of the Annual Exclusion Feature and should be carefully considered as to how the parameter settings are applied. Please contact your Client Services team for more information on this.

4.1.4.1 Annual Exclusion Values Explained

For every screening session, one of the following values must be provided for this parameter:

- "1" = Annual Exclusion Feature is not used
- "2" = Annual Exclusion Feature is used
- "3" = Two results will be provided; one using the Annual Exclusion Feature and one not

4.1.4.2 Annual Exclusion Parameter Examples

Single Trip Policy = "\$G_IsAnnual": 1,

AMT Policy without AEF = "\$G_IsAnnual": 1,

AMT Policy with AEF = "\$G_IsAnnual": 2,

4.1.5 Winter Sports Exclusion Feature

VRR has identified several conditions that are not suitable for cover whilst undertaking winter sports activities and the Winter Sports Exclusion Feature (WSEF) allows VRR's clients to exclude these from cover. However, VRR recognises that not all clients want to use this feature; please find specific instructions on how to use or not use this feature below.

Parameter: "\$G_IsWinterSport"

Type: Integer

Status: Mandatory parameter required for all screenings

4.1.5.1 Winter Sports Exclusion Values Explained

For every screening session, one of the following values must be provided for this parameter:

- "1" = Winter Sports Exclusion Feature **is not** used
- "2" = Winter Sports Exclusion Feature **is** used
- "3" = Two results will be provided: one using the Winter Sports Exclusion Feature and one not

4.1.5.2 Annual Exclusion Parameter Examples

Policy with WSEF = "\$G_IsWinterSport": 1,

Policy without WSEF = "\$G_IsWinterSport": 2,

4.1.6 Coronavirus (COVID-19)

RRT3 Travel users are now able to declare Coronavirus (COVID-19) and some related conditions. However, VRR recognises that not all clients want their customers to declare these conditions; therefore, by default, Coronavirus (COVID-19) is hidden in the database. For clients wanting their customers to declare these conditions, it is, therefore, their responsibility to expose these conditions; please find specific instructions on how to expose these conditions below.

Parameter: "\$G_ShowClasses"

Type: Comma Separated String

Status: Optional – not required for all screenings

Value: "6"

The "\$G_ShowClasses" parameter applies to all class overrides, which include pregnancy, pregnancy complications and Covid-19 related medical conditions. This can be seen in the screening parameter section (4.1).

4.1.6.1 Coronavirus (COVID-19) Parameter Example

For clients who do not want to expose Coronavirus (COVID-19) and its related conditions, neither this parameter nor any value needs to be provided.

For clients who do want to expose Coronavirus (COVID-19) and its related conditions, please include the following in the register screening ticket:

"\$G_ShowClasses": 6,

RRT3 Travel Exclusion Flags

By using the Annual Exclusion Feature, the requested exclusions feature and the Winter Sports Exclusion Feature, some of the customer's declared conditions may be excluded from cover. RRT3 communicates this by raising an exclusion flag for each affected condition.

4.1.7 Where are the exclusion flags raised?

In the result, each <Condition> node will have an <exclusionType> sub-node; this is where RRT3 raises an exclusion flag for that specific condition.

Following a completed screening, the client's system will need to query every <exclusionType> sub-node to search for exclusions.

4.1.8 Exclusion Flags

The below table shows all the possible exclusion flags in RRT3 Travel alongside a brief description.

Exclusion Flag	Description
"None"	This is the default value and means that this condition has not been excluded
"Derived"	This condition has been excluded because another condition in its group has been excluded
"Requested"	This condition has been excluded at the client (insurer)'s request
"WS"	This condition has been excluded by the Winter Sports Exclusion Feature
"AMT"	This condition has been excluded by the Annual Exclusion Feature
"ADT"	This condition has been excluded by the Annual Exclusion Feature

Please note that "None" is the default value and means this condition has not triggered an exclusion. If there is any value other than "None" returned in the <exclusionType> node, then this condition has been excluded.

Whilst both "AMT" and "ADT" are possible exclusion flags for the Annual Exclusion Feature, only "ADT" is currently used.

It is possible for multiple exclusion flags to be raised for a single condition, e.g. <exclusionType> WS Derived </exclusionType>.

Important: Please note that you will be required to review the results received from the RRT to ascertain medical condition exclusion outcomes.

5 Results

Risk Rating Tool Messages

Information Message: To help answer some questions, RRT3 provides supporting information in the JSON Response; this text is provided in the "InfoMessage" node, which is a sub-node of the Question node.

Warning Messages: At certain points in the customer journey, pop-up warning messages are provided by RRT3 to be presented to the customer. RRT3 provides these messages in the JSON Response; specifically, in the "ResponseMessage" node of the JSON Response.

Screening Results

Screening results can be returned either in XML or JSON format.

5.1.1 Travel XML Results

```
<Screening>
  <ScreeningPath>
    <!--
First published in 2013. All rights reserved. (c) Verisk Risk Rating Ltd. This XML
data is provided subject to the condition that it shall not be used, re-
used, or otherwise circulated without the publisher's prior consent in any format
or schema other than that in which it is published and without a similar condition
including this condition being imposed upon the subsequent user. For the avoidance
of doubt the XML data in its original format and schema represents the result of a
Verisk Risk Rating Screening which may only be used in its entirety, and it may not
be re-used or otherwise circulated by extracting a part, or parts, or by reformatting
in any way.-->
    <ScreeningSettings>
      <Age>24</Age>
      <Gender>Male</Gender>          <isAMT>>false</isAMT>
      <isWinterSport>>false</isWinterSport>
      <regionId>1</regionId>
      <LinkedCondition>0</LinkedCondition>
      <IsRiskRatingLite>>false</IsRiskRatingLite>
    </ScreeningSettings>
    <MedicalRisk>1.4</MedicalRisk>
    <ScreeningDate>22/03/2021</ScreeningDate>
    <CustomerJourney>
      <CustomerAction>
```

```

        <Type>Condition declared</Type>
        <ActionDateTime>13/10/2020 04:26:04</ActionDateTime>
        <Description>Screening started for declared condition 'High blood
pressure'</Description>
    </CustomerAction>
    <CustomerAction>
        <Type>Complication declared</Type>
        <ActionDateTime>13/10/2020 04:26:08</ActionDateTime>
        <Description>Forwarded into condition 'Cholesterol levels'</Descr
ption>
    </CustomerAction>
    <CustomerAction>
        <Type>Condition saved</Type>
        <ActionDateTime>13/10/2020 04:26:13</ActionDateTime>
        <Description>Finished declaring condition(s) 'High blood pressure
, Cholesterol levels'</Description>
    </CustomerAction>
</CustomerJourney>
<ScreeningHistory>
    <conditions>
        <Condition>
            <name>High blood pressure</name>
            <ICD9>401.9</ICD9>
            <ICD>I10</ICD>
            <TSF>False</TSF>
            <RiskRatingLite>False</RiskRatingLite>
            <Score>1.1</Score>
            <id>1332</id>
            <GroupId>1</GroupId>
            <GroupScore>1.4</GroupScore>
            <exclusionType>None</exclusionType>
            <questions>
                <QandA>
                    <question>How many medicines does your doctor advise
you to take for high blood pressure?</question>
                    <answer>0</answer>
                </QandA>
                <QandA>
                    <question>Have you ever been a smoker?</question>
                    <answer>No</answer>
                </QandA>
                <QandA>
                    <question>Have you been advised to take medication to
lower your cholesterol?</question>
                    <answer>Yes</answer>
                </QandA>
            </questions>
        </Condition>
    </conditions>
</ScreeningHistory>
</CustomerJourney>

```

```

        </QandA>
    </questions>
</Condition>
<Condition>
    <name>Cholesterol levels</name>
    <ICD9>272.0</ICD9>
    <ICD>E78.0</ICD>
    <TSF>False</TSF>
    <RiskRatingLite>False</RiskRatingLite>
    <Score>1.3</Score>
    <id>1330</id>
    <ParentId>1</ParentId>
    <GroupScore>1.4</GroupScore>
    <exclusionType>None</exclusionType>
    <questions>
        <QandA>
            <question>Has a blood test EVER at any time shown you
r cholesterol level to be raised?</question>
            <answer>Yes</answer>
        </QandA>
        <QandA>
            <question>Have you ever been a smoker?</question>
            <answer>No</answer>
        </QandA>
    </questions>
</Condition>
</conditions>
</ScreeningHistory>
<SystemData>
    <ScreeningData>H4sIAAAAAAEEAN1W227bRhD9lcw+6EUMxKsuL0Eiu7EL1Gkjt0FQ9G
G40xQXXnLV3aVcI/C/d0iKMmOp6SVAgYQPhDicy86ZM0f8yN9atVU16I2wiLWqt1fgSr7iaZwtMrGUwVx
mYZAUsQwWKYpgsSyKJSShXEaCT/nzsNnhCs7chovC3miTg/4RLFT0tTq+qhutp/wdup2pHf6AzsEWB/Nr
EHfN7gKFpgivGMU0Gg/PH3q8etH/saaZnfTVDlavgqn/Nq9E17tKXEB2mFrWJtqp+kQkq+8bci0NrVUT
ym0j6M0fR2UGwQrylu01TWfh9liyn+u1e8NtmisjaQ6fA4A4SzCQCwSDJI5iGAZYRrEk0WJT0cQ5kCI/N
Sgeyo6PI1qDqbhhx12hle1u0d7AR6G9k+6e7sbUv42+B/ye9h+Ast3xt6DlSNcDpYWHgI0109ZN6VptHy
N6xLFHY5iWgzCKK0ENzTXlg38kQq/B9syhJ6pz+u6MM+mS7WUdf7oNk43i8Pkm07K3LMK6gdWoVRC1eiY
NHR7MI2lX8Iby0Du1cPWxLxhHu6QFWQu1bZkuTZGsp2l8o3F1/xxeop0/LUgnWaz9IjMjfkbpPmmMnf0i
qla6EYSaKh8iZZZ3DbEZUYYOSMUaNYQfKZgLC60goAORJgZ2Qj/gt2WODgQo9WWtsB7ykbhkhz3sqMKkn0
ltPB0mqT0D1+WhVy/454cdp0/UuYJ9P0Xc0ylzEhkGXR605weX/G+D62Xiv84tni3mxy4/oPviFYnT6BS
1HrBuF+RxEbq16dSxNWlDvfa7I0jqjCTq0Rr88Pc/1H0QZR1F8NaPePHjs9uSSdBG6tg+IdKeJR6c5Xbpu
dCeSGv2VpM7OCGoyx2WR0Z9S0Mc4SKI8DpZFmgRFLnMp42gh4/DbEdRs9m/Ywm9L5Wj0UDs26TZH1f2It
SrQqwonn9/BOBrJLW0uH0ZH0+7Z5S+X7xj4ThraXMyV5r4+oRDTVfM3FMuRWwGpeH5lo69lCGKwht+c1m
b/UGu/UA5mxzLr5xTp2dz5HZ0uWMAmZ4Rj0v1uUK0gb7XxN1rPknXjvKnQfk9krPGhNz/+CSCBWOLgCgA
A</ScreeningData>
    <ScreeningType>Travel</ScreeningType>

```

```

    <ScreeningHash>53686c9d-7d61-4f3d-85ec-89ff9a41d92c</ScreeningHash>
    <OriginalScreeningHash>53686c9d-7d61-4f3d-85ec-
89ff9a41d92c</OriginalScreeningHash>
    <UtcScreeningDateTime>2021-03-22T10:08:49.412Z</UtcScreeningDateTime>
    <BlackBoxModel>Internal Data specific to BB3</BlackBoxModel>
  </SystemData>
</ScreeningPath>
</Screening>

```

5.1.2 Cancellation XML Results

```

<Screening>
  <ScreeningPath>
    <!--
First published in 2013. All rights reserved. (c) Verisk Risk Rating Ltd. This XM
L data is provided subject to the condition that it shall not be used, re-
used, or otherwise circulated without the publisher's prior consent in any format
or schema other than that in which it is published and without a similar conditi
on including this condition being imposed upon the subsequent user. For the avoid
ance of doubt the XML data in its original format and schema represents the resul
t of a Verisk Risk Rating Screening which may only be used in its entirety, and i
t may not be re-
used or otherwise circulated by extracting a part, or parts, or by reformatting i
n any way.-->
    <ScreeningSettings>
      <Age>24</Age>
      <Gender>Male</Gender>          <isAMT>false</isAMT>
      <isWinterSport>false</isWinterSport>
      <regionId>1</regionId>
      <LinkedCondition>0</LinkedCondition>
      <IsRiskRatingLite>false</IsRiskRatingLite>
    </ScreeningSettings>
    <CancellationRisk>1</CancellationRisk>
    <ScreeningDate>22/03/2021</ScreeningDate>
    <CustomerJourney>
      <CustomerAction>
        <Type>Condition declared</Type>
        <ActionDateTime>13/10/2020 04:26:04</ActionDateTime>
        <Description>Screening started for declared condition 'High blood
pressure'</Description>
      </CustomerAction>
      <CustomerAction>
        <Type>Complication declared</Type>
        <ActionDateTime>13/10/2020 04:26:08</ActionDateTime>

```

```

        <Description>Forwarded into condition 'Cholesterol levels'</Description>
    </CustomerAction>
    <CustomerAction>
        <Type>Condition saved</Type>
        <ActionDateTime>13/10/2020 04:26:13</ActionDateTime>
        <Description>Finished declaring condition(s) 'High blood pressure
, Cholesterol levels'</Description>
    </CustomerAction>
</CustomerJourney>
<ScreeningHistory>
    <conditions>
        <Condition>
            <name>High blood pressure</name>
            <ICD9>401.9</ICD9>
            <ICD>I10</ICD>
            <TSF>False</TSF>
            <RiskRatingLite>False</RiskRatingLite>
            <Score>1</Score>
            <id>1332</id>
            <GroupId>1</GroupId>
            <GroupScore>1</GroupScore>
            <exclusionType>None</exclusionType>
            <questions>
                <QandA>
                    <question>How many medicines does your doctor advise
you to take for high blood pressure?</question>
                    <answer>0</answer>
                </QandA>
                <QandA>
                    <question>Have you ever been a smoker?</question>
                    <answer>No</answer>
                </QandA>
                <QandA>
                    <question>Have you been advised to take medication to
lower your cholesterol?</question>
                    <answer>Yes</answer>
                </QandA>
            </questions>
        </Condition>
        <Condition>
            <name>Cholesterol levels</name>
            <ICD9>272.0</ICD9>
            <ICD>E78.0</ICD>
            <TSF>False</TSF>

```



```

        <RiskRatingLite>False</RiskRatingLite>
        <Score>1</Score>
        <id>1330</id>
        <ParentId>1</ParentId>
        <GroupScore>1</GroupScore>
        <exclusionType>None</exclusionType>
        <questions>
            <QandA>
                <question>Has a blood test EVER at any time shown you
r cholesterol level to be raised?</question>
                <answer>Yes</answer>
            </QandA>
            <QandA>
                <question>Have you ever been a smoker?</question>
                <answer>No</answer>
            </QandA>
        </questions>
    </Condition>
</conditions>
</ScreeningHistory>
<SystemData>
    <ScreeningData>H4sIAAAAAAAAEAN1W227bRhD9lcw+6EUMxKsuL0Eiu7EL1Gkjt0FQ9G
G40xQXXnLV3aVcI/C/d0iKMmOp6SVAgYQPhDicy86ZM0f8yN9atVU16I2wiLWqt1fgSr7iaZwtMrGUwVx
mYZAUsQwWKYpgsSyKJSShXEaCT/nzsNnhCs7chovC3miTg/4RLFT0tTq+qhutp/wdup2pHf6AzsEWB/Nr
EHfN7gKFpgivYGMU0Gg/PH3q8etH/saaZnfTVDlavgqn/Nq9E17tKXEB2mFrWJtqp+kQkq+8bci0NrVUT
ym0j6M0fR2UGwQrylu01TWfH9liyn+u1e8NtmisjaQ6fA4A4SzcQCwSDJI5iGAZYRrEk0WJT0cQ5kCI/N
Sgeyo6PI1qDqbhhx12hle1u0d7AR6G9k+6e7sbUv42+B/ye9h+Ast3xt6DlSNcDpYWHgI0109ZN6VptHy
N6xLFHY5iWgzCKK0EnzTXlg38kQq/B9syhJ6pz+u6MM+mS7WUdf7oNk43i8Pkm07K3LMK6gdWoVRC1eiY
NHR7MI2lX8Iby0Du1cPWxLxhHu6QFWQu1bZkuTZGsp2l8o3F1/xxeop0/LUgnWaz9IjMjfkbpPmmMnf0i
qla6EYSaKh8iZZZ3DbEZUYY0SMUaNYQfKZgLC60goAORJgZ2Qj/gt2WODgQo9WWtsB7ykbhkhz3sqMKkn0
ltPB0mqT0D1+WhVy/454cdp0/UuYJ9P0Xc0ylzEhKXGR605weX/G+D62Xiv84tni3mxy4/oPviFYnT6BS
1HrBuF+RxEbq16dSxNWlDvfa7I0qjCTq0Rr88Pc/1H0QZR1F8NaPePHjs9uSSdBG6tg+IdKeJR6c5Xbpu
dCeSGv2VpM70CGoyx2WR0Z9S0Mc4SKI8DpZFmgRFLnMp42gh4/DbEdRs9m/Ywm9L5Wj0UDs26TZH1f2It
SrQqwonn9/BOBrJLW0uH0ZH0+7Z5S+X7xj4ThraXMyV5r4+oRDTVfM3FMuRWwGpeH5lo69lCGkWh+c1m
b/UGu/UA5mxzLr5xTp2dz5HZ0uWMAmZ4Rj0v1uUK0gb7XxN1rPknXjvKnQfk9krPGhNz/+CSCBW0LgCgA
A</ScreeningData>
    <ScreeningType>Cancellation</ScreeningType>
    <ScreeningHash>53686c9d-7d61-4f3d-85ec-89ff9a41d92c</ScreeningHash>
    <OriginalScreeningHash>53686c9d-7d61-4f3d-85ec-
89ff9a41d92c</OriginalScreeningHash>
    <UtcScreeningDateTime>2021-03-22T10:08:49.413Z</UtcScreeningDateTime>
    <BlackBoxModel>Internal Data specific to BB3</BlackBoxModel>
</SystemData>
</ScreeningPath>
</Screening>

```

5.1.3 Travel JSON Results

```
{
  "Screening": {
    "ScreeningPath": {
      "ScreeningSettings": {
        "regionId": 1
      },
      "MedicalRisk": 1.32,
      "ScreeningDate": "11/01/2023",
      "ScreeningHistory": {
        "conditions": [
          {
            "name": "High blood pressure",
            "ICD9": "401.9",
            "ICD": "I10",
            "TSF": "False",
            "RiskRatingLite": "False",
            "exclusionType": "None",
            "questions": [
              {
                "question": "How many medicines does your doctor advise you to take for high blood pressure?",
                "answer": [
                  "1"
                ]
              },
              {
                "question": "Has your dose been increased or have you been prescribed a new tablet in the last 6 months?",
                "answer": [
                  "Yes"
                ]
              },
              {
                "question": "Have you ever been a smoker?",
                "answer": [
                  "No"
                ]
              }
            ]
          }
        ]
      }
    }
  }
}
```

```
"question": "Have you been advised to take medication to lower your cholesterol?",
    "answer": [
        "No"
    ]
}
]
}
],
"SystemData": {
    "ScreeningData":
"H4sIAAAAAAACTVVS2/bOBD+KwQvvliF9bBr51KKTh8p0DRbpyiKYg800bKIUKSWD7tGkf++Q8lS1CTY7GH3UB8EczzfD7OMOf9JOVO6mZ2nALoKXevWeuomd0wZcZzBZlUixTlhQ5Ewlj5Tx5WQqxFEU5F8WKTunDsNnpIzzx6X8Y9hlKSka5XIV6C5ae6aDUlL5TZsvUNbOsBg/W9fbP4BqjHXwE59g0evNrxm9DcwFcYYSX6DEKCMr3q189vv+k76wJ TZ86ndJLd8693CNwyZSDaFiBuLFyHKBn3gY0rY0W8h5iWiI5gujwgNsAsr27A1pcYni6WU/pFy78CRJLWRmAewqxESyTVKslTyJNiUbBkuV1tE5GV2XyR5UWOb1P6RwB3n7RfjXL2pptjA63hXLsD2AvmWX/8R6f71PSQf/b+J3zPdjl2FgPfGntgVox4OVkiPUjoVt2jbioTlHgN6wr4LYxiIgdpNl9N6RXqimdP6R0m/spsvDi4xnNe6tIM6tJrBcwBkZqrIIAwfSQ1CmlbCYmmCdHE4hnt0C8ITXTGEe2yhHBGoswwcKUyJIEhf+SpsPbg5XlkRykr2K8JcJwb+yLmN+9ldb5oaZx7bM8LYba35tDzHcqSGpwiIKfER5hyi8xxVui6aos0VzJXfwgyFf0bvpy1ux/lDX9L2XN0tXLgZlv4J7T9ZxMNByQEuzgJ0gh044wEm0jfvt9OiviYD9J0P4FaQt4JF5UORKNFue08SSy+qygq+xeTzZoh5JtcZjFa2xfjfREkisb2nZLtVtSNW7nFva7A7igYgrcSiGLOkwWpjfaVe1rb/Hfrdr6YzQeWrsexz0m5qc4tbfC86AigUWGjhF3AoRiqd4ZIPEpbNU7ZN7TAoEoi8GhE4inyDLJ4cIOFyh7J6j2gYPtmzBJNMun6LwnMTtCcoYMTBred0z+fpSPeTroBToROxtThgnxu+F1mbT5bLv69ch3Z/8TauFv2o27o5pwYhtx4RhuiDJ616y1eGYXUGTUKyX1Yz5sfEGUCrtGzGR7NM4+pSE/e4AMbb0f/Anfv5KNqHg/UFr11HLwuSEImT7hOWt8NKOD4zI+f946TdXDDe1GA/4Ak0HDvz3d90/c1kmglkAAA==",
    "ScreeningType": "Travel",
    "ScreeningHash": "6c82e06f-481a-43ad-aaf5-7fdd8d4f5d49",
    "OriginalScreeningHash": "6c82e06f-481a-43ad-aaf5-7fdd8d4f5d49",
    "UtcScreeningDateTime": "2023-01-11T12:25:02.418Z",
    "BlackBoxModel": "Internal Data specific to BB3"
}
}
```

5.1.4 Cancellation JSON Results

```
{
  "Screening": {
    "ScreeningPath": {
      "ScreeningSettings": {
```

```
        "regionId": 1
    },
    "CancellationRisk": 1,
    "ScreeningDate": "11/01/2023",
    "ScreeningHistory": {
        "conditions": [
            {
                "name": "High blood pressure",
                "ICD9": "401.9",
                "ICD": "I10",
                "TSF": "False",
                "RiskRatingLite": "False",
                "exclusionType": "None",
                "questions": [
                    {
                        "question": "How many medicines does your doctor
advise you to take for high blood pressure?",
                        "answer": [
                            "1"
                        ]
                    },
                    {
                        "question": "Has your dose been increased or have
you been prescribed a new tablet in the last 6 months?",
                        "answer": [
                            "Yes"
                        ]
                    },
                    {
                        "question": "Have you ever been a smoker?",
                        "answer": [
                            "No"
                        ]
                    },
                    {
                        "question": "Have you been advised to take
medication to lower your cholesterol?",
                        "answer": [
                            "No"
                        ]
                    }
                ]
            }
        ]
    }
},
```

```

"SystemData": {
  "ScreeningData":
"H4sIAAAAAAACTVVS2/bOBD+KwQvvlIF9bBr51KkTh8p0DRbpyiKYg800bKIUKSWD7tGkf++Q81S1CTY
7GH3UB8EczjzzfD70M0f9JOV06mZ2nALoKXevWeuomd0wZcZzBZlUixTlhQ5Ewlj5Tx5WQqxFEU5F8WKT
unDsNnpLzzx6X8Y9h1KsKA5XIV6C5ae6aDU1L5TZsvUNb0sBg/W9fbP4BqjHXwE59g0evNrxm9DcwFcYY
SX6DEKCMr3q189vv+k76wJTZ86ndJLd8693CNwyZSDaFibulFYhKBn3gY0rY0W8h5iWI5gujwgNsAsr27
A1pcYni6WU/pFy78CRJLWRmAeWqxEsYTVKsITyJNiUbBkuV1tE5GV2XyR5UWOb1P6RwB3n7RfjXL2pptj
A63hXLsD2AvmWX/8R6f71PSQf/b+J3zPdJ12FgPfGntgVox40VkiPUjoVt2jbioTlHgN6wr4LYxiIgdPn
l9N6RXqimdP6R0m/spsvDi4xnNe6tIM6tJrBcwBkZqrIIAwfSQ1CM1bCYmvmCdHE4hnt0C8ITXTGEe2yh
hBGoswwcKUYJIEHf+SpsPbg5X1kRykr2K8JcJwb+yLmN+9ldb5oaZx7bM8LYba35tDzHcqSGpwiIKfER5
hYi8xXVui6aos0VzJXfWgyFf0bvpY1ux/1DX9L2XN0tXLgZ1v4J7T9ZxMNBvYQEUzgJ0gh044wEm0jfVt9
OiviYD9J0P4FaQt4JF5U0RKNfUe08SSy+qygq+xeTzZoh5JtcZjFa2fjfREkisb2nZLtvSNW7nFva7A7
igYgrcSiGLOkwWpjfaVe1rb/HfRdr6YzQeWrsxz0m5qc4tbfC86AigUWGJhF3AoRiqd4ZIpEpBnU7ZN7T
AoEoi8GhE4inyDLJ4cIOFyh7J6j2gYPTmzBjNMun6LWnMTtCcoYMTBred0z+fpSPeTroBToR0XtThgnxa
u+F1mbT5bLv69cH3Z/8TauFv2o27o5pwYhtx4RhuiDJ616y1eGYXUgTUKyX1Yz5sfeGUcRtGzGR7NM4+p
sE/e4AMbb0f/AnfV5KNqHg/UFr11HLwuSEImT7h0Wt8NKOD4zI+f946TdXDe1GA/4Ak0HDvz3d90/c1kM
gkAAA==",
    "ScreeningType": "Cancellation",
    "ScreeningHash": "6c82e06f-481a-43ad-aaf5-7fdd8d4f5d49",
    "OriginalScreeningHash": "6c82e06f-481a-43ad-aaf5-7fdd8d4f5d49",
    "UtcScreeningDateTime": "2023-01-11T12:25:02.418Z",
    "BlackBoxModel": "Internal Data specific to BB3"
  }
}

```

Result Output Description

The output result, either in XML or JSON format, is composed by the following fields:

Field Name	Value	Notes
\Screening	Root node	
\Screening\ScreeningPath	Container	Container holding data on screening results for a set of screening parameters.
\Screening\ScreeningPath\ScreeningSettings	ScreeningSettings	Container for screening settings data
\Screening\ScreeningPath\MedicalRisk	Decimal	Medical Risk score
\Screening\ScreeningPath\CancellationRisk	Decimal	Cancellation Risk Score

\Screening\ScreeningPath\ScreeningDate	Date	Date when the screening result was generated
\Screening\ScreeningPath\LinkedConditions	Container	Container listing the complete set of linked conditions for the screening. There can be 0 or 1 of these elements
\Screening\ScreeningPath\LinkedConditions\LinkedCondition	LinkedCondition	Linked condition element. There can be 1 or multiple of these elements
\Screening\ScreeningPath\ScreeningHistory	Container	
\Screening\ScreeningPath\ScreeningHistory\conditions	Container	Container for all conditions that were declared in the screening
\Screening\ScreeningPath\ScreeningHistory\conditions\Condition	Condition	Condition element contains all result data related to the condition. There can be 1 or multiple of these elements
\Screening\ScreeningPath\CustomerJourney	Container	Container for all customer actions that we logged during the screening session. There can be 0 or 1 of these elements
\Screening\ScreeningPath\CustomerJourney\CustomerAction	CustomerAction	CustomerAction element contains data related to single customer action. There can be 0, 1 or multiple of these elements
\Screening\SystemData	String	For internal RRT3 usage

Screening Settings

Field Name	Value	Notes
\ScreeningSettings	Root node	
\ScreeningSettings\regionId	String	Region ID values vary depending on integration location. Please contact your VRR Client Services Representative to confirm the appropriate values for your integration.
\ScreeningSettings\isAMT	Integer	Policy Type (Single, AMT) True = Annual Multi Trip False = Single Trip Both = Both

\ScreeningSettings\LinkedCondition	Integer	<p>0 = Linked Conditions are not provided</p> <p>1 = Directly Linked Conditions only are provided</p> <p>2 = Indirectly Linked Conditions only are provided</p> <p>3 = Both Directly and Indirectly Linked Conditions are provided</p>
\ScreeningSettings\IsWS	Integer	Is this a winter sport policy: True = Yes False = No Both = Both
\ScreeningSettings\Age	Integer	The age if provided on the Register Screening and if the value "Age" is set on the \$G_ResultDisplayParameters string array parameter
\ScreeningSettings\Gender	String	The gender if provided on the Register Screening and if the value "Gender" is set on the \$G_ResultDisplayParameters string array parameter

Condition

Field Name	Value	Notes
\Condition	Root node	
\Condition\name	String	Condition Name
\Condition\ICD9	String	Condition ICD9
\Condition\ICD	String	Condition ICD10
\Condition\Score	Decimal	Condition Medical Score
\Condition\TSF	Boolean	Condition Time Sensitive Flag
\Condition\RiskRatingLite	Boolean	Condition Risk Rating Lite Flag
\Condition\GroupID	Integer	Condition Group ID

\Condition\GroupScore	Decimal	Condition Group Score
\Condition\UniqueHashCode	String	UUID to uniquely identify a condition. This is an optional feature.
\Condition\questions	Container	Container listing all questions and answers data for the condition
\Condition\questions\QandA	Container	Container for question and selected answer(s) data. There can be 0, 1 or multiple of these elements
\Condition\questions\QandA\question	String	Question text
\Condition\questions\QandA\answer	String	Answer text. There can be 1 or multiple of these elements
\Condition\questions\QandA\negativeanswer	String	Answer text that was not selected by the user. There can be 0, 1 or multiple of these elements
\Condition\LinkedConditions	Container	Container listing linked conditions associated with the current condition for screening. There can be 0 or 1 of these elements
\Condition\LinkedConditions\LinkedCondition		Linked condition element. There can be 1 or multiple of these elements

Multiple Language Output

The Multiple Language Output is an optional feature where screening results are provided in more than a single language.

When calculating the assessment score, every item inside the Result array property from the RRT3 API response contains a language property. Normally this Language property matches the language configured on your account, but using this feature, additional results with additional languages will be provided.

Important: Please note that this feature will produce multiple outputs depending on the secondary languages configured in the user account.

The Language property value is RFC-5646 compliant.

The elements in the result that will be translated when possible are:

- Condition Names
- Questions
- Answers

An example of the Result property returned from BB3 can be found below:

```
"Result": [  
  {  
    "CalculationType": "1",  
    "Result": "<Screening>...</Screening>",  
    "Language": "da"  
  },  
  {  
    "CalculationType": "2",  
    "Result": "<Screening>...</Screening>",  
    "Language": "da"  
  },  
  {  
    "CalculationType": "1",  
    "Result": "<Screening>...</Screening>",  
    "Language": "en"  
  },  
  {  
    "CalculationType": "2",  
    "Result": "<Screening>...</Screening>",  
    "Language": "en"  
  }  
],
```

6 API

Introduction

Risk Rating Tool contains additional functionality that is not included in the widget, such as rescoring an existing screening or migrating from a legacy screening. This functionality requires using the RRT3 API, which is what the widget uses internally.

RRT3 API is a stateless API. From its perspective, each request is completely independent of the other. However, a screening process is a stateful and incremental process, where the user would perform multiple sequential actions, such as declaring a condition and then answering questions about it. The state for this is provided as part of each request, modified by RRT3 API as needed, and returned in the response. The

client system would then read it and react accordingly (for example, updating its UI to show that a question has been answered and a new one has appeared), attaching this updated state to the next request once the user completes an action.

To avoid inconsistent or erroneous behaviour, the state should not be modified by the client system and should be included in the requests as it was provided by RRT3 API.

Authentication

RRT3 API expects all requests to be properly authenticated. The authentication mechanism is the same as described in Section 3.8.1:

A temporary authentication token is obtained after providing gateway credentials. This token expires in one hour and should be periodically renewed.

A licence key is also provided by VRR which does not expire.

These mechanisms are not alternatives. Each request must include both to be successfully authenticated. The temporary authentication token must be provided in an `Authorization` header using a `Bearer` token. The licence key must be provided in a `x-vrr-auth` header.

A properly authenticated request would look like:

```
curl '<BB3 API ENDPOINT>' \  
  -X POST \  
  -H 'Authorization: Bearer <TEMPORARY TOKEN>' \  
  -H 'x-vrr-auth: <LICENSE KEY>' \  
  -H 'Content-Type: application/json' \  
  -d '<BODY>'
```

Please refer to Section 3.7.1 for further information on how to retrieve the temporary access token.

Recalculate/ Rescore

Given a previous Screening Result, the method recalculates the Medical Risk Score.

POST <https://gatewayuat.verisk.com/rating/au/travel/calculation/rescore/v2>
(APAC)

Body

`$G_Age`

`$G_DirectlyLinked` See 4.1 for detailed information on screening parameters.

`$G_IndirectlyLinked`

`$G_Regions`

`$G_Username`

`$G_ScreeningData`

Headers

<code>Authorization</code>	<code>Bearer <GATEWAY_CREDENTIALS></code>	See the authentication section.
<code>x-vrr-auth</code>	<code><License Key></code>	See the authentication section.
<code>Content-Type</code>	<code>application/json</code>	

A sample `/rescore` request:

```
curl 'https://gatewayuat.verisk.com/rating/au/travel/calculation/rescore/v2' \
-X POST \
-H 'Authorization: Bearer <TEMPORARY TOKEN>' \
-H 'x-vrr-auth: <LICENSE KEY>' \
-H 'Content-Type: application/json' \
-d '{
  "$G_Username": "<YOUR USERNAME>",
  "$G_Age": 30,
  "$G_DirectlyLinked": false,
  "$G_IndirectlyLinked": false,
  "$G_Regions": "<Value>",
  "$G_IsAnnual": 1,
  "$G_IsWinterSport": 1,
  "$G_ScreeningData": "<PREVIOUS RESULT>"
}'
```

The above command returns JSON structured like this:

```
{
  "ResponseMessage": null,
  "Result": [
    {
```

```
    "CalculationType": "1",
    "Language": "en",
    "Result": "<RESCORED RESULT>"
  }
]
```

`ResponseMessage` is the result message. If `null`, rescore was successful; otherwise, it contains the error message.

`CalculationType` will be 1 for travel, 2 for cancellation.

`Result` is the body of the result.

`Language` is the RFC-5646 compliant code.

Rescreen

Given a previous Screening Result, the register method processes the earlier screening result.

During rescreen, as a result of possible changes in the Risk Rating Tool content since the original screening has been made, conditions may be forced to be edited or screened from scratch. In this case, the list of Declared Conditions will be annotated with warning messages and these Declared Conditions will require editing or rescreening to complete the Rescreen and submit for Scoring. The Declared Conditions section handles this when required.

POST <https://gatewayuat.verisk.com/rating/au/travel/registration/register/v2>
(APAC)

Body

`$G_Age`

`$G_DirectlyLinked`

`$G_IndirectlyLinked`

`$G_Regions`

`$G_Username`

`$G_ScreeningData`

See 4.1 for detailed information on screening parameters.

Headers

Authorization	Bearer <GATEWAY_CREDENTIALS>	See the authentication section.
x-vrr-auth	<License Key>	See the authentication section.
Content-Type	application/json	

A sample rescreen request:

```
curl 'https://gatewayuat.verisk.com/rating/au/travel/calculation/rescore/v2' \
-X POST \
-H 'Authorization: Bearer <TEMPORARY TOKEN>' \
-H 'x-vrr-auth: <LICENSE KEY>' \
-H 'Content-Type: application/json' \
-d '{
  "$G_Username": "<YOUR USERNAME>",
  "$G_Age": 30,
  "$G_DirectlyLinked": false,
  "$G_IndirectlyLinked": false,
  "$G_Regions": "<Value>",
  "$G_IsAnnual": 1,
  "$G_IsWinterSport": 1,
  "$G_ScreeningData": "<PREVIOUS RESULT>"
}'
```

The above command returns JSON structured like this:

```
{
  "OriginalScreeningHash": "00000000-0000-0000-0000-000000000000",
  "ScreeningHash": "00000000-0000-0000-0000-000000000000",
  "GlobalParameters": {
    "$G_Regions": "608",
    "$G_DirectlyLinked": false,
    "$G_Username": "<YOUR USERNAME>",
    "$G_Locale": "en",
    "$G_IndirectlyLinked": false,
    "$G_Age": 30
  },
  "ResponseMessage": ""
}
```

```
"BackupDeclarations": null,  
"Result": null,  
"Declarations": [  
  {  
    [OMITTED FOR BREVITY]  
    [SHOULD CONTAIN THE SAME DECLARATION IN $G_ScreeningData]  
  }  
],  
"SelectedConditions": [],  
"CustomerJourney": []  
}
```

7 Customization

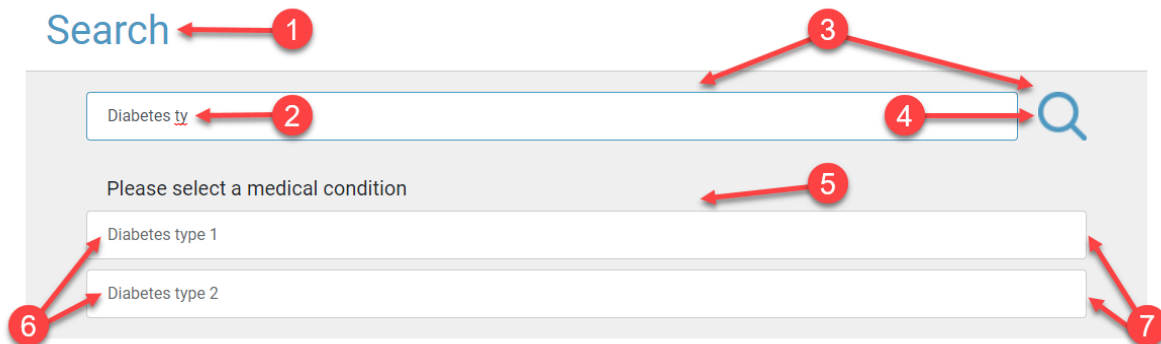
The widget can be customized by redefining the default stylesheet provided by Verisk - BootstrapVerisk.css. This was developed based on an isolated version of Bootstrap, by redefining all relevant style classes using the *Verisk-* prefix.

Please take care to avoid conflicts with your custom CSS and the Verisk stylesheets (e.g. Z-Index).

The styles for the main component are displayed below.



1. `divclass="row no-gutters veriskcollapsible-header"`
2. `divclass="questionItem"`
3. `divclass="verisk-answer"`
4. `button type="button" class="verisk-btn verisk-btn-secondary verisk-btn-edit"`
5. `divclass="row no-gutters verisk-box-notactive verisk-mt-10"`
6. `divclass="row no-gutters verisk-box-active verisk-mt-10"`
7. `label class="verisk-btn verisk-btn-secondary verisk-light mb-0 verisk-answer-button"`
8. `<button type="button" class="verisk-btn verisk-btn-secondary verisk-btn-cancel"`

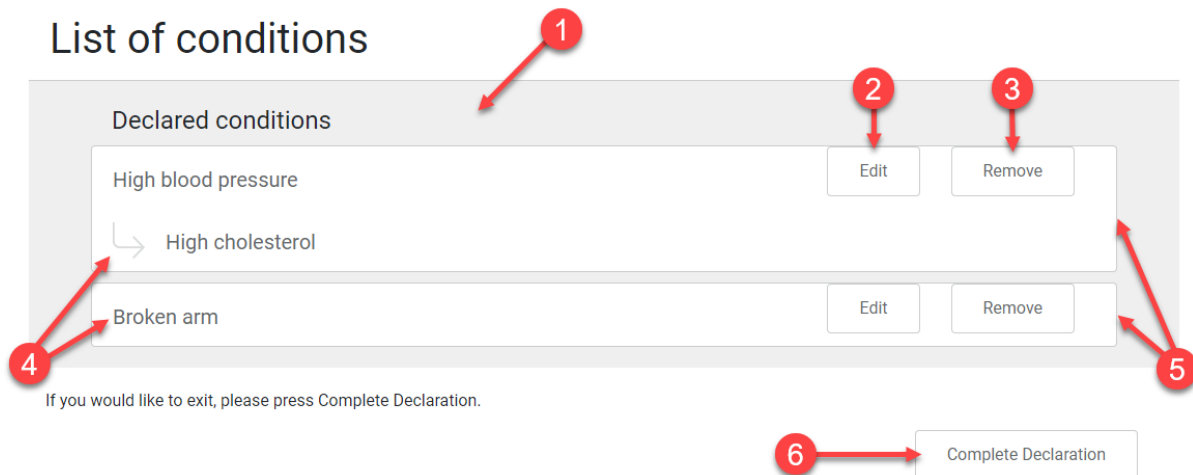


The image shows a search form with the following elements:

- 1. Search title
- 2. Input field containing "Diabetes ty"
- 3. Search button (magnifying glass icon)
- 4. Search button (magnifying glass icon)
- 5. Search results list
- 6. Search results list item "Diabetes type 1"
- 7. Search results list item "Diabetes type 2"

1. `<h1 class="verisk-focused verisk-title">Search</h1>`
2. `input class="verisk-form-control"`
3. `form class="verisk-searchBox verisk-background verisk-top-border verisk-pt-20"`
4. `div class="verisk-btn-search-active verisk-ml-20-8-5"`
5. `div class="searchResultList verisk-form-group"`
6. `button type="button" class="verisk-btn verisk-btn-secondary verisk-btn-left btn-block verisk-light"`
7. `div class="verisk-searchResultItem verisk-mt-10 verisk-ml-60-5 verisk-mr-60-5"`

List of conditions



The image shows a form titled "List of conditions" with the following elements:

- 1. Declared conditions header
- 2. Edit button
- 3. Remove button
- 4. Declaration item row
- 5. Declaration item row
- 6. Complete Declaration button

If you would like to exit, please press Complete Declaration.

1. `div class="declarationList verisk-background verisk-top-border verisk-pb-20"`
2. `button type="button" class="verisk-btn verisk-btn-secondary verisk-btn-edit"`
3. `button type="button" class="verisk-btn verisk-btn-secondary verisk-right-float verisk-btn-remove"`
4. `div class="declarationItem row"`
5. `div class="verisk-border rounded verisk-light verisk-declarationGroup verisk-mt-10 verisk-ml-60-5 verisk-mr-60-5"`
6. `button type="button" id="btnReactFinish" class="verisk-btn verisk-btn-secondary verisk-right-float verisk-btn-finish"`

8 Encryption

General information

8.1.1 Purpose

RRT3 Widget is a client-side application providing customers with an easier integration path into the Travel Risk Rating Tool. The RRT3 Widget allows having a working screening system without the need to develop a complex user interface to the underlying API.

There is an initial communication between the customer and Verisk, in the form of obtaining credentials (such as the gateway token). Afterwards, all communications are handled directly between the user's browser and Verisk servers. At the end of a screening process, the client-side application is responsible for processing the results as per the customer needs. This will usually involve sending the results over to the server for further processing, such as generating an insurance premium.

However, because the client-side application is executing on the user's browser, there could be a risk of a user modifying the requests and responses in any way, ultimately providing the customer with a modified screening result.

To mitigate the risk of a user modifying the RRT3 requests and/or responses, an encryption system has been implemented, involving both encryption and signature. This prevents the end user from viewing them directly, which increases the safety of the customer processes and decisions, as well as the safety of the rating outcome by preventing the user from modifying the results.

8.1.2 Overview

Result encryption is a mandatory feature for RRT3 Travel and is applied to every account. Once encryption has been activated for your account, all the screening results (the content returned by the calculate endpoint) will be encrypted and signed.

The decrypted content can be accessed via two methods: decrypt the content on your servers or perform a request to our servers to have the content decrypted.

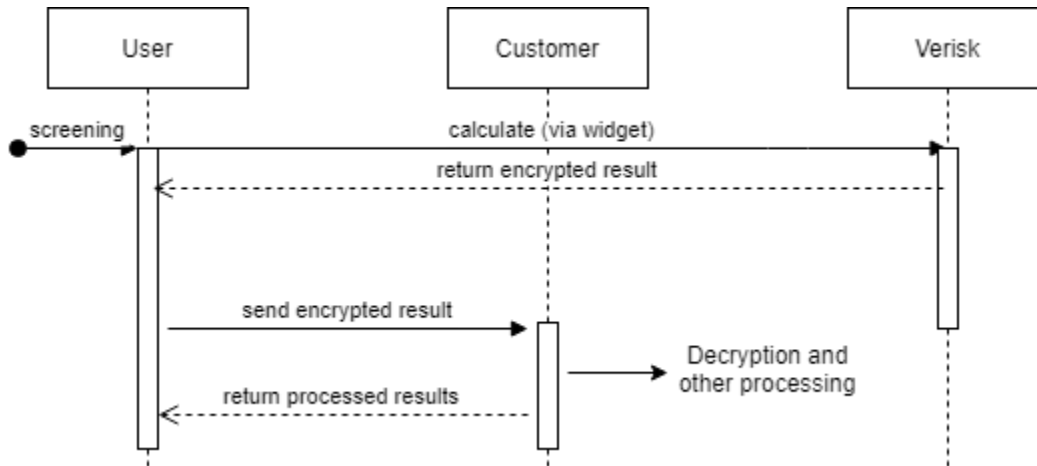
Implementing the decryption mechanism in your system is the recommended approach, as it will allow you more flexibility whilst also being faster.

8.1.3 Implementing decryption

If you opt to follow the recommended approach of decrypting the content on your own system, you will need to use a Verisk-provided encryption key. This is a passphrase that can be used to both decrypt and verify the results. It is very important that this passphrase is kept safe and is not shared with anyone,

especially not with the users. Please refer to the technical details (Section 8.3) of this guide for further information on how to implement this decryption process.

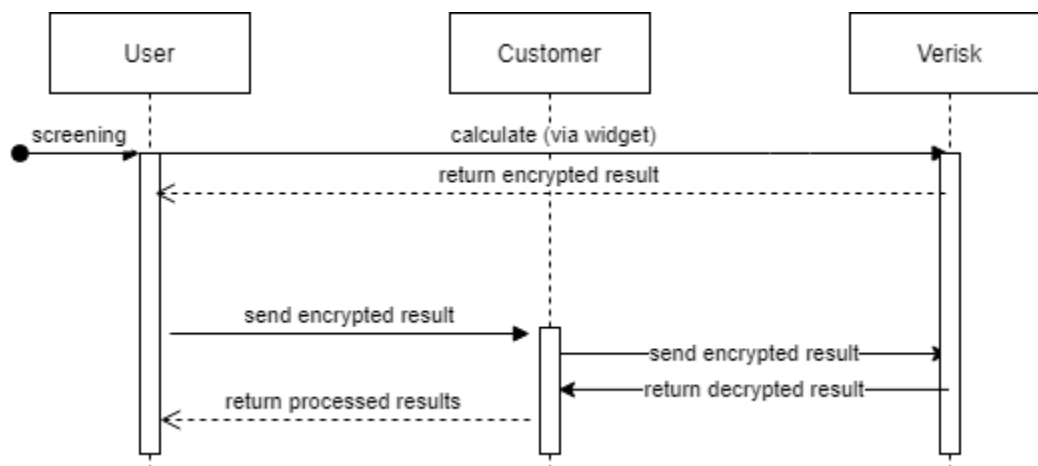
With this approach, when a screening is completed on your client-side application, you can send it to your servers to be decrypted.



8.1.4 Using a decryption endpoint

The other option available to you is relying on our system for decryption. When you receive an encrypted result, you should then perform a call to our decryption endpoint, which will decrypt the result and send it back to you.

This call should not be performed from a client-side application, but from your own servers; otherwise, the security objective would not be achieved. To call this decryption endpoint you will need to use an additional Encryption Licence Key that will be provided to you by Verisk. It is important that you do not share this licence key with the user or your client-side application.



Technical Details

8.1.5 Encryption and authentication

The screening results are first encrypted using AES-256-CBC; a widely used encryption algorithm, which is considered very secure. It has easy-to-implement libraries on most platforms. This ensures that the content of the screening can only be read by a client application with the proper key. The provided ciphertext will contain an initial vector, or IV, in its first 16 bytes, and the rest will be the plaintext being encrypted. Please refer to the structure of the message section, as well as the samples, for further information.

Afterwards, these encrypted screening results are signed using HMAC-SHA26. This digital signature assures that the results have not been modified. This is, therefore, an Encryption-Then-Authenticate process.

8.1.6 Key Derivation

Our client services will provide you with a passphrase. For the decryption process, you will need to derive two keys from that password: a key for encryption, and a key for authentication. Two keys are used for security reasons, as a single key for both usages is considerably less secure.

For deriving the two keys, you will need your passphrase (as provided by Verisk), as well as two salts. These two salts are generated randomly for each operation and included as part of the message. They are unencrypted, as they are not required to be kept secret, especially because they will never be reused.

The method used for key derivation is PBKDF2, with 1000 iterations, and SHA-1 as a digest. This is widely supported in most platforms. The number of iterations has been chosen to balance the computational time required while having proper security. As the passphrase is very strong, with 415 bits of entropy (and 3.6e162 possible combinations) there is no need to use a large number of iterations.

8.1.7 Structure of the message

Reserved	MAC	Key ID	Encryption Salt	Authentication Salt	Ciphertext
----------	-----	--------	-----------------	---------------------	------------

- Reserved: A 64-bit (8 bytes) reserved header. This is for internal use by Verisk and can be safely ignored by customers.
- MAC: A 256-bit (32 bytes) message authentication code. This is used to verify that the content has not been modified.
- Key ID: A 128-bit (16 bytes) UUID that identifies the passphrase that was used to derive the keys for encryption and authentication. This is provided in case the key needs to be rotated for any reason.

- Encryption Salt: A 64-bit (8 bytes) random value is used to derive the encryption key. This is generated randomly for each message.
- Authentication Salt: A 64-bit (8 bytes) random value is used to derive the authentication key. This is generated randomly for each message.
- Ciphertext: A variable-length field that contains the encrypted screening results. The 16 initial bytes on this will be the initial vector, to be provided to the AES-256 decryptor.

This whole sequence of bytes will be encoded as base64 for transmission over the wire.

8.1.8 Decryption process

The input as returned from Verisk will be a base64 string that contains the sequence of bytes as described above. This will be sent to the widget and handled as described on our documentation, so you should be able to capture it and send it to your servers.

The decryption process consists of several steps:

- Partitioning of the message to retrieve the required information.
- Derivation of encryption and authentication keys.
- Verification of the integrity of the message.
- The decryption of content.

Once the encrypted result is on your server, the decryption process can start. The first step would be to decode the base64 into a byte array or an equivalent structure available on your platform. Afterwards, you need to slice it, according to the structure of the message, so you have the different pieces that compose it: the MAC, the salts, the ciphertext, etc.

Once the encryption and authentication salts have been retrieved, you will be able to derive the keys. By using the PBKDF2 as described in Section 2.2, and together with your Verisk-provided passphrase, you can obtain the keys that were used to sign and encrypt the message.

When the keys have been derived, the first step would be to verify the signature of the message, that is, verify that the message has not been modified by the user. For verification, we use the HMAC-256 algorithm, as described in Section 2.1, by using the key derived from the authentication salt.

The content that has been signed is the combination of Key ID, Encryption Salt, Authentication Salt and Ciphertext. That is, everything starting from byte #32 to the total length of the message. You need to calculate the HMAC-256 for that combination of bytes and compare it against the provided MAC (the first 32 bytes). Most platforms will allow you to directly perform this comparison as part of the “verify” method. If the message has been modified, because the MAC that you calculated does not match the one provided, you should consider the screening as fraudulent.

The last step, assuming that the message was verified, is to decrypt the content. You need to further slice the ciphertext: the first 16 bytes are the initial vector (IV), and the rest are the actual content. You need to provide these two parts to your AES-256-CBC implementation, together with your derived encryption key. After this, you will get back an array of bytes that you can turn into a UTF8 string, to get the result.

Please, refer to our samples section to see the actual code implementing this whole process.

8.1.9 Decryption endpoint

The second approach is relying on our system to perform the decryption process. We have added a new endpoint that receives the encrypted result as an input and returns the decrypted result as an output.

This is a private endpoint, intended to be used only from your system and not directly invoked by the end-user. To ensure this, it requires a new set of credentials in the form of a new Encryption Licence Key, provided by Verisk. For security reasons, you need to ensure that this licence key is never shared with the end-user.

POST <https://gatewayuat.verisk.com/rating/au/travel/calculation/decrypt/v2>
(APAC)

POST <https://gatewayuat.verisk.com/rating/uk/travel/calculation/decrypt/v2>
(EU)

Body

<code>EncryptedXml</code>	The encrypted result as returned from RRT3 API. Please note that initially, all the RRT3 results were in XML format but now it supports results in JSON format as well. That's the reason why this property is called EncryptedXml although it should be called EncryptedResult. It can't be changed because it will generate a breaking change.
---------------------------	--

Headers

<code>Authorization</code>	<code>Bearer <GATEWAY_CREDENTIALS></code>	This is the Gateway Licence Key
<code>x-vrr-auth</code>	<code><Internal License Key></code>	This is the Encryption Licence Key
<code>Content-Type</code>	<code>application/json</code>	

A sample `/decrypt` request:

```
curl 'https://gatewayuat.verisk.com/rating/au/travel/calculation/decrypt/v2' \
-X POST \
-H 'Authorization: Bearer <TEMPORARY TOKEN>' \
-H 'x-vrr-auth: <INTERNAL LICENSE KEY>' \
-H 'Content-Type: application/json' \
-d '{
  "EncryptedXml": "<YOUR ENCRYPTED DATA>"
}'
```

The above command returns JSON structured like this:

```
{  
  "DecryptedXml": "<DECRYPTED RESULT>"  
}
```

`DecryptedXml` is the body of the result, decrypted.

Rescreen and rescore

There are several use cases where you need to provide information about a previous screening as part of the operation: if you are performing a rescore, or if you are performing a rescreen. Sometimes, this screening will not have even been handled by your server yet, such as if you allow some kind of “review” functionality, where screening may have been completed but your process is still ongoing, and the user should be able to review or modify their declarations.

For all these cases, our endpoints that receive a previous screening (such as rescreen or rescore) also support receiving encrypted data. If we receive an encrypted result as part of the screening data parameter, we will decrypt it as appropriate and handle all the processing transparently. A rescored result will be returned encrypted, as will a rescreened result, once that new session has been completed.

We strongly recommend against allowing a user to see the actual screening result, so if you need to provide them with a previous screening that has already been processed by your server (for example, a rescreen or pause/resume functionality), please use the encrypted version instead of the decrypted one.

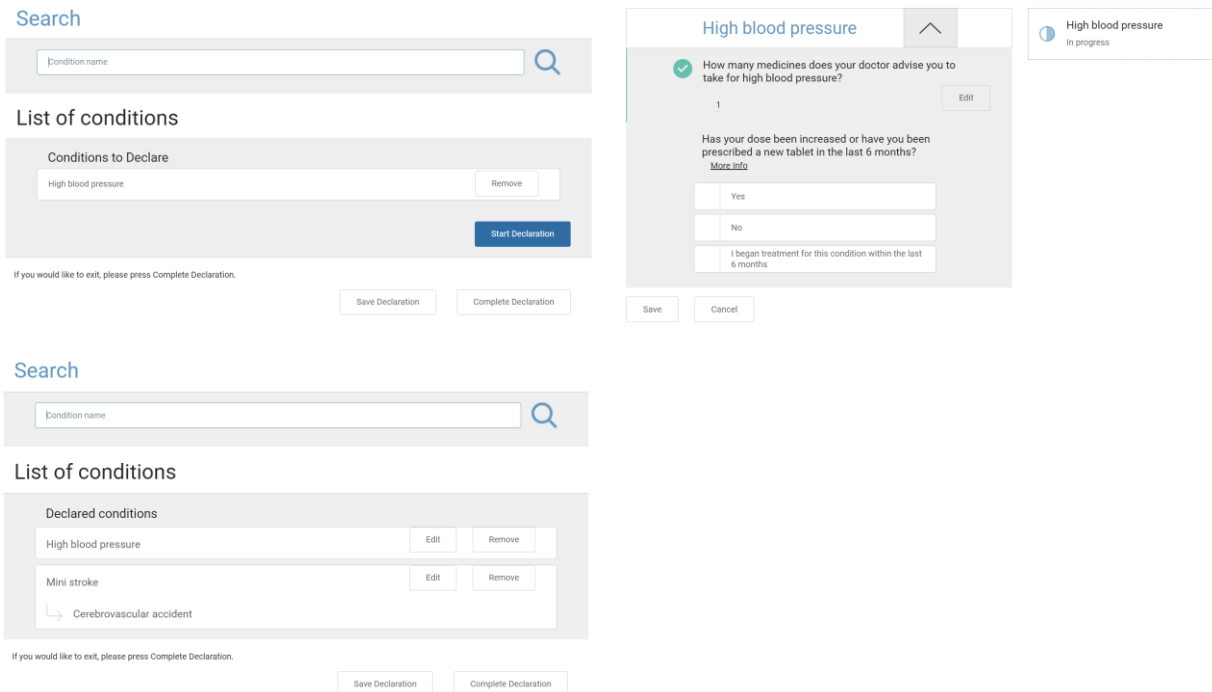
9 Save/Resume Screening

General information

The Save and Resume screening is an optional feature, and it is not required to be enabled to be able to perform an assessment in the RRT3 Widget or RRT3 API.

9.1.1 Save Screening

RRT3 Widget allows the user to save a screening. The screening can be saved at any moment during the screening process: when declaring conditions, during Q&A or when all the declarations are completed but the 'Complete Declaration' button wasn't clicked.

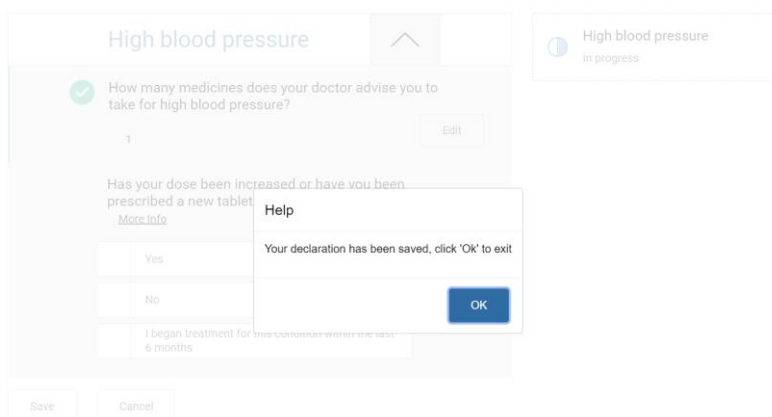


The left screenshot shows the 'List of conditions' section with a search bar and a list of conditions to declare, including 'High blood pressure'. The right screenshot shows the 'High blood pressure' declaration form with questions about medicines and treatment, and a 'Save Declaration' button.

The screening will be saved with the current date and the response from RRT3 API will be a base64 UTF-8 Encoded string in case the user doesn't have encryption enabled or an encrypted string result in case the user has encryption enabled. The saved screening will have a validity of 90 days from the time it was saved. For resuming the declaration, we expect the input as-is, so please do not transform it in any way.

The result from the save screening feature can only be captured when the variable `EVENT_LISTENERS` is configured with an `onFinish` function.

The save screening will end the interaction with the Widget and the `onFinish` event allows for the user to take back control to take any desired action. The `onFinish` event will be triggered after the user clicks the button 'OK' from the modal dialog.



The screenshot shows a modal dialog box with the text 'Your declaration has been saved, click 'Ok' to exit' and an 'OK' button.

9.1.2 Resume Screening

RRT3 Widget also allows the user to resume a previously saved screening. If the parameter `$G_PartialScreening` is supplied with a value when initializing the Widget, a previously saved partial screening will be restored at the point where it was saved. This flow will always take precedent over a register flow, so if the intention is to register a new screening, the `$G_PartialScreening` parameter shouldn't be supplied. The value to set in the `$G_PartialScreening` is the value from the response from the Save Screening action described above.

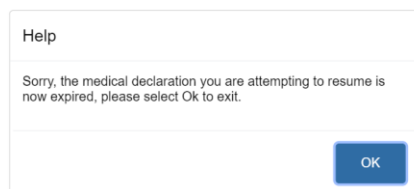
If user has Include Policy Information feature disabled, and mandatory parameters are not sent in the `$G_Parameters` object, RRT3 API will return an error.

To restore a saved screening, the date when the screening was saved will be evaluated and a popup message will be shown to the user in case the screening has expired. This will trigger the `onError` event when the user clicks the 'OK' button which will allow the user to take back control and perform any action it deems necessary.

There are also invalid requests or unexpected errors returned from RRT3 API which will not show any popup but will trigger the `onError` event finishing all user interaction with the Widget and giving back control to the user allowing any action to be executed.


In case there are any data changes supported by the RRT3 API between the moment when the screening was saved and the resume screening is performed, a warning message will be displayed in the Widget for the user to edit the declaration that contains any data change.

When the `RESUME_SCREENING_EXPIRED` error code is triggered, the modal dialog displayed to the user will look as follows:



When there is a data change between the time when the screening was saved and the time the resume is triggered, the widget will behave as shown below:

Search

Condition name 

List of conditions

Conditions to Declare

High blood pressure

Declared conditions

We need further information regarding this condition. Please edit this condition

High blood pressure

If you would like to exit, please press Complete Declaration.

Technical Details

9.1.3 Save Screening

When saving a screening the RTT3 Widget will execute a request to the RRT3 API to the `/save/v2` endpoint.

POST <https://gateway.verisk.com/rating/uk/travel/registration/save/v2> (UK)

POST <https://gateway.verisk.com/rating/au/travel/registration/save/v2> (APAC)

Body

Screening Context

Headers

Authorization	Bearer <GATEWAY_CREDENTIALS>	See the authentication section.
---------------	------------------------------	---------------------------------

x-vrr-auth	<License Key>	See the authentication section.
------------	---------------	---------------------------------

Content-Type	application/json
--------------	------------------

A sample save request:

```
curl 'https://gateway.verisk.com/rating/uk/travel/registration/save/v2' \
-X-POST \
-H 'Authorization: Bearer <TEMPORARY TOKEN>' \
```

```
-H 'x-vrr-auth: <LICENSE KEY>' \  
-H 'Content-Type: application/json' \  
-d '{  
  "OriginalScreeningHash": "48fc9bcc-8f75-4f5c-bd92-500e46a9b957",  
  "ScreeningHash": "48fc9bcc-8f75-4f5c-bd92-500e46a9b957",  
  "GlobalParameters": {  
    "$G_Username": "<YOUR USERNAME>",  
    "$G_TripDuration": 366,  
    "$G_DirectlyLinked": false,  
    "$G_Regions": "1",  
    "$G_ShowExtendedInformation": "true",  
    "$G_IsAnnual": 2,  
    "$G_IndirectlyLinked": false,  
    "$G_Locale": "en",  
    "$G_Age": 21,  
    "$G_IsWinterSport": 1,  
    "$G_LeadTime": 3  
  },  
  "ResponseMessage": "",  
  "BackupDeclarations": "",  
  "Result": null,  
  "Declarations": [  
    {  
      "GroupNumber": 1,  
      "IsActive": false,  
      "IsCompleted": true,  
      "Conditions": [  
        {  
          "ConditionNumber": 1,  
          "DeclaredSearchTermId": 1488,  
          "UniqueHashCode": "60915a9c-179d-4e4b-9c62-b695bd3a2419",  
          "Questions": [],  
          "Warning": "",  
          "Exclusion": 0,  
          "State": 3,  
          "Editable": true,  
          "Id": 2240,  
          "Name": "Hyperactivity"  
        }  
      ],  
      "Id": 1,  
      "Name": "D - 'Hyperactivity'"  
    }  
  ],  
  "SelectedConditions": [],  
}
```

```
"CustomerJourney": [
  {
    "TimeStamp": 1636627238,
    "Parameters": [
      "Hyperactivity"
    ],
    "CustomerActionType": 1,
    "Id": 0,
    "Name": null
  },
  {
    "CustomerActionType": 5,
    "TimeStamp": 1636627249,
    "Parameters": [
      "Hyperactivity"
    ]
  }
]
```

The above command returns JSON structured like this:

```
{
  "Result": "<BASE64 ENCODED STRING. OMITTED FOR SIMPLICITY>",
  "ResponseMessage": null
}
```

As for the result that will be provided to the `onFinish` event `result` argument will be a base64 encoded string as follows (just a sample):

[illegible]

In case an error occurs, an error array will be provided in the `onError` event `errors` argument:

```
[
  {
    code: "SAVE_SCREENING_INVALID",
    detail: "Invalid request data",
  }
]
```

9.1.4 Resume Screening

When resuming a screening the RTT3 Widget will execute a request to the RRT3 API to the `/resume/v2` endpoint.

POST <https://gateway.verisk.com/rating/uk/travel/registration/resume/v2> (UK)

POST <https://gateway.verisk.com/rating/au/travel/registration/resume/v2> (APAC)

Body

`$G_Username`

`$G_PartialScreening`

Headers

Authorization	Bearer <GATEWAY_CREDENTIALS>	See the authentication section.
x-vrr-auth	<License Key>	See the authentication section.
Content-Type	application/json	

A sample resume request:

```
curl 'https://gateway.verisk.com/rating/uk/travel/registration/resume/v2' \
-X-POST \
-H 'Authorization: Bearer <TEMPORARY TOKEN>' \
-H 'x-vrr-auth: <LICENSE KEY>' \
-H 'Content-Type: application/json' \
-d '{
  "$G_Username": "<YOUR USERNAME>",
```

```
"$G_PartialScreening": "<BASE64 ENCODED STRING WITH A SAVED DECLARATION>"
}
```

The above command returns JSON structured like this:

```
{
  "OriginalScreeningHash": "48fc9bcc-8f75-4f5c-bd92-500e46a9b957",
  "ScreeningHash": "48fc9bcc-8f75-4f5c-bd92-500e46a9b957",
  "GlobalParameters": {
    "$G_Username": "VRRTRavel",
    "$G_TripDuration": 366,
    "$G_DirectlyLinked": false,
    "$G_Regions": "1",
    "$G_IsAnnual": 2,
    "$G_IndirectlyLinked": false,
    "$G_Locale": "en",
    "$G_Age": 21,
    "$G_IsWinterSport": 1,
    "$G_LeadTime": 3
  },
  "ResponseMessage": "",
  "BackupDeclarations": "",
  "Result": null,
  "Declarations": [
    {
      "GroupNumber": 1,
      "IsActive": false,
      "IsCompleted": true,
      "Conditions": [
        {
          "ConditionNumber": 1,
          "DeclaredSearchTermId": 1488,
          "UniqueHashCode": "60915a9c-179d-4e4b-9c62-b695bd3a2419",
          "Questions": [],
          "Warning": "",
          "Exclusion": 0,
          "State": 3,
          "Editable": true,
          "Id": 2240,
          "Name": "Hyperactivity"
        }
      ],
      "Id": 1,
      "Name": "D - 'Hyperactivity'"
    }
  ]
}
```

```
],
"SelectedConditions": [],
"CustomerJourney": [
  {
    "TimeStamp": 1636627238,
    "Parameters": [
      "Hyperactivity"
    ],
    "CustomerActionType": 1,
    "Id": 0,
    "Name": null
  },
  {
    "CustomerActionType": 5,
    "TimeStamp": 1636627249,
    "Parameters": [
      "Hyperactivity"
    ]
  }
]
}
```

The above response won't be passed to any configured event and will restore a declaration to the point where it was previously saved.

In case there is as an error, an error array will be provided in the `onError` event `errors` argument:

```
[
  {
    code: "RESUME_SCREENING_EXPIRED",
    detail: "Resume data is expired"
  }
]
```

10 Migration

Given a previous legacy Screening Result (pre-RRT3), this method migrates legacy screening to the RRT3 format and then recalculates Score. Migration functionality is provided via a POST request in 2 different endpoints, `/registration/register/v2` and `/calculation/rescore/v2`.

Migration is only possible for Travel screening type (Cancellation is not supported).

10.1.1 Register

POST <https://gatewayuat.verisk.com/rating/au/travel/registration/register/v2>
(APAC)

POST <https://gatewayuat.verisk.com/rating/uk/travel/registration/register/v2>
(UK)

Body

\$G_Age

\$G_DirectlyLinked

\$G_IndirectlyLinked

\$G_Regions

See 4.1 for detailed information on screening parameters.

\$G_Username

\$G_TripDuration

\$G_LeadTime

\$G_IsAnnual

\$G_IsWinterSports

\$G_IsScreeningLite

Headers

Authorization

Bearer <GATEWAY_CREDENTIALS>

See the authentication section.

x-vrr-auth

<License Key>

See the authentication section.

Content-Type

application/json

A sample /resgister request:

```
curl 'https://gatewayuat.verisk.com/rating/au/travel/calculation/rescore/v2' \
```

```
-X POST \  
-H 'Authorization: Bearer <TEMPORARY TOKEN>' \  
-H 'x-vrr-auth: <LICENSE KEY>' \  
-H 'Content-Type: application/json' \  
-d '{  
  "$G_Username": "<YOUR USERNAME>",  
  "$G_Age": 30,  
  "$G_DirectlyLinked": false,  
  "$G_IndirectlyLinked": false,  
  "$G_Regions": "<Value>",  
  "$G_IsAnnual": 1,  
  "$G_IsWinterSport": 1,  
  "$G_IsScreeningLite": 0  
}'
```

The above command returns JSON structured like this:

```
{  
  "OriginalScreeningHash": "00000000-0000-0000-0000-000000000000",  
  "ScreeningHash": "00000000-0000-0000-0000-000000000000",  
  "GlobalParameters": {  
    "$G_Regions": "608",  
    "$G_DirectlyLinked": false,  
    "$G_Username": "<YOUR USERNAME>",  
    "$G_Locale": "en",  
    "$G_IndirectlyLinked": false,  
    "$G_Age": 30  
  },  
  "ResponseMessage": "",  
  "BackupDeclarations": null,  
  "Result": null,  
  "Declarations": [  
    {  
      [OMITTED FOR BREVITY]  
    }  
  ],  
  "SelectedConditions": [],  
  "CustomerJourney": []  
}
```

10.1.2 Rescore

POST <https://gatewayuat.verisk.com/rating/uk/travel/calculation/rescore/v2>
(UK)

POST <https://gatewayuat.verisk.com/rating/au/travel/calculation/rescore/v2>
(APAC)

Body

\$G_Age	Age in years at time of coverage (Optional)	25
\$G_IsWinterSport	Determine whether this is a Winter Sport policy or not	1
\$G_DirectlyLinked	List of condition(s) directly linked to the Condition(s)true declared	
\$G_IndirectlyLinked	List of condition(s) indirectly linked to the Condition(s) declared	false
\$G_Regions	String (in quotes)	"608"
\$G_IsAnnual	Determines whether this is a Single Trip or Annual Multi Trip Policy	1
\$G_Username	(Mandatory)	
\$G_ScreeningData	Entire XML of a legacy screening, encoded as base64.	
\$G_TripDuration		
\$G_LeadTime		
\$G_IsScreeningLite		

Headers

Authorization	Bearer <GATEWAY_CREDENTIALS>	See the authentication section.
x-vrr-auth	<License Key>	See the authentication section.
Content-Type	application/json	

A sample /rescore request:

```
curl 'https://gatewayuat.verisk.com/rating/uk/travel/calculation/rescore/v2' \
-X POST \
-H 'Authorization: Bearer <TEMPORARY TOKEN>' \
-H 'x-vrr-auth: <LICENSE KEY>' \
-H 'Content-Type: application/json' \
```

```
-d '{
  "$G_Username": "<YOUR USERNAME>",
  "$G_Age": 30,
  "$G_TripDuration": 4,
  "$G_LeadTime": 0,
  "$G_IsWinterSport": 1,
  "$G_IsAnnual": 1,
  "$G_DirectlyLinked": false,
  "$G_IndirectlyLinked": false,
  "$G_Regions": "1",
  "$G_ScreeningData": "<LEGACY XML>",
  "$G_IsScreeningLite": 0
}'
```

The above command returns JSON structured like this:

```
{
  "ResponseMessage": null,
  "Result": [
    {
      "CalculationType": "1",
      "Result": "<RESCORED RESULT>",
      "Language": "en"
    },
    {
      "CalculationType": "2",
      "Result": "<RESCORED RESULT>",
      "Language": "en"
    }
  ]
}
```

`ResponseMessage` is the result message. If `null`, migration was successful; otherwise, it contains the error message.

`CalculationType` will be 1 for travel, 2 for cancellation.

`Result` is the body of the result.

`Language` is the RFC-5646 compliant code.

11 Frequently Asked Questions

1. Is the RRT3 widget compatible with single-page applications?

Yes, it is developed on React + Redux SPA frameworks and is compatible with single-page applications.

2. Are there primary and secondary URLs for RRT3?

With RRT3 there is no requirement for a primary or secondary URL as it provides a stateless solution. In the unlikely event of a server outage, the customer or users will be seamlessly switched to another Verisk server and continue the declaration with no interruption. Furthermore, there is no need for IT Teams to manually switch to another server.

3. What ID values should be parsed for \$G_Regions?

The ID values for \$G_Regions changes depending on the location of the integration, e.g. UK-hosted Region IDs are different to Australia-hosted Region IDs. For confirmation on Region ID values, please contact your VRR Client Services Representative.

4. When a user has selected multiple answers to multiple selection questions, how can these all be extracted into a post-purchase endorsement document?

Extract all child nodes of the <QandA> node. The child nodes include:

- <Question> - this node displays the question displayed to the user
- <Answer> - this node displays the answer(s) selected by the user
- <negativeanswer> - this node displays an answer that the user did not select in a multiple selection question

5. How does RRT3 determine the language of the dataset?

When a new RRT3 account is created, it will have a language setting set based on the region and local language of the client. Any requests made by the Widget will be done using English language, however the response from the RRT3 will be in the language set on the RRT3 account.

6. How are the screening result XMLs encoded?

The response from the calculation endpoint is, as every other endpoint in this API, a UTF-8 encoded JSON. This response contains, among other things, the result of the calculation.

Inside the result and depending on your account's configuration, will be:

- **For the XML output:** an XML string whose one particular thing about is that it is formatted with spaces and Windows new line characters ("r\n")

- For the JSON output: a simple JSON object.

7. What is the Original Screening Hash and the Screening Hash used for?

The Screening Hash is used to uniquely identify a screening.

The Original Screening Hash is used to group a set of screenings that can be done using the Rescore endpoint or when Rescreening.

A screening will start with the Original Screening Hash with the same value as the Screening Hash, for example, "c1b5d23e-3c58-4bfe-9204-d28ed5fca6cd". When a request is done to the /calculate endpoint the value will remain the same, but if a request to the /rescore endpoint is executed or the \$G_ScreeningData parameter is used, the Original Screening Hash will remain with the value of "c1b5d23e-3c58-4bfe-9204-d28ed5fca6cd" but the Screening Hash will have a new Guid string value generated.

8. Should validations be applied to the response JSON Object or to the XML structure?

Your system can validate the results and responses obtained from RRT. However, when validating or otherwise processing the results, we recommend that you do so only for the existing fields, and do not disallow the addition of new fields. For example, adding a new field to the JSON response or a new node to an XML structure.

As the system evolves and new features are implemented, these responses may have slight changes. Verisk will not remove or replace a field name or type, but new fields may be added regularly (both to our JSON and our XML responses).

It is strongly advised that your system should be capable to deal with this situation for example, not performing a strict validation that rejects newly added fields, to ensure the maximum compatibility.

12 Samples

Decryption

12.1.1 .NET

```
using System;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;

static void Main(string[] args)
```

```
{
    var data = "<Encrypted result goes in here>";
    var key = "<Your passphrase goes in here>";
    Console.WriteLine(Decrypt(key, data));
}

private static string Decrypt(string password, string base64Message)
{
    var message = Convert.FromBase64String(base64Message);

    // Start at index 8 (first 8 bytes are internally used by Verisk)
    const int veriskHeaderLength = 8;
    var index = 0 + veriskHeaderLength;

    // Extract the authentication tag.
    var mac = new byte[32];
    Array.Copy(message, index, mac, 0, mac.Length);
    index += mac.Length;

    // Extract the key ID tag.
    var key = new byte[16];
    Array.Copy(message, index, key, 0, key.Length);
    index += key.Length;

    // Extract the encryption salt value.
    var encryptionSalt = new byte[8];
    Array.Copy(message, index, encryptionSalt, 0, encryptionSalt.Length);
    index += encryptionSalt.Length;

    // Extract the authentication salt value.
    var authenticationSalt = new byte[8];
    Array.Copy(message, index, authenticationSalt, 0, authenticationSalt.Length);
    index += authenticationSalt.Length;

    // Extract the ciphertext to be decrypted.
    var ciphertext = new byte[message.Length - index];
    Array.Copy(message, index, ciphertext, 0, ciphertext.Length);

    // Extract the payload that will be authenticated.
    var payload = new byte[key.Length + encryptionSalt.Length + authenticationSalt.Length + ciphertext.Length];
    Array.Copy(message, veriskHeaderLength + mac.Length, payload, 0, payload.Length);

    // Derive the encryption key using PBKDF2.
```

```
byte[] encryptionKey;
using (var generator = new Rfc2898DeriveBytes(password, encryptionSalt, 1000)
)
{
    encryptionKey = generator.GetBytes(32);
}

// Derive the authentication key using PBKDF2.
byte[] authenticationKey;
using (var generator = new Rfc2898DeriveBytes(password, authenticationSalt, 1000))
{
    authenticationKey = generator.GetBytes(32);
}

// Authenticate the message to verify lack of tampering.
var hash = Hash(authenticationKey, payload);
var isEqual = mac.SequenceEqual(hash);
if (!isEqual)
{
    throw new Exception("The message could not be authenticated");
}

// Finally, decrypt the ciphertext with the desired algorithm.
var decryptedData = Decrypt(encryptionKey, ciphertext);

var plainText = Encoding.UTF8.GetString(decryptedData);
return plainText;
}

private static byte[] Decrypt(byte[] key, byte[] input)
{
    using (var aes = Aes.Create("AES"))
    {
        // The first 16 bytes on the input are the IV.
        // This IV is randomly generated by the sender for each message.
        // We need to separate the IV from the message for proper decoding.
        var iv = new byte[aes.IV.Length];
        Array.Copy(input, 0, iv, 0, iv.Length);

        var payload = new byte[input.Length - aes.IV.Length];
        Array.Copy(input, iv.Length, payload, 0, payload.Length);

        aes.Key = key;
        aes.IV = iv;
    }
}
```

```
        aes.Mode = CipherMode.CBC;
        aes.Padding = PaddingMode.PKCS7;
        using (var decryptor = aes.CreateDecryptor())
        using (var msDecrypt = new MemoryStream(payload))
        using (var csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
        {
            using (var msOutput = new MemoryStream())
            {
                csDecrypt.CopyTo(msOutput);

                return msOutput.ToArray();
            }
        }
    }

private static byte[] Hash(byte[] key, byte[] input)
{
    using (var hmac = HMAC.Create("HMACSHA256"))
    {
        hmac.Key = key;
        return hmac.ComputeHash(input);
    }
}
```

12.1.2 Python

Requires packages `pyaes` and `pbkdf2`.

```
import base64, uuid, sys
import pyaes, pbkdf2
import hmac, hashlib

def verify(key, data, verification):
    processor = hmac.new(key, msg=data, digestmod=hashlib.sha256)
    return hmac.compare_digest(processor.digest(), verification)

def decrypt(key, data):
    iv = data[:16]
    payload = data[16:]

    decrypter = pyaes.Decrypter(pyaes.AESModeOfOperationCBC(key, iv))
    plain = decrypter.feed(payload)
    plain += decrypter.feed()
```

```
    return plain.decode()

data = base64.b64decode("<Encrypted result goes in here>")
password = "<Passphrase goes in here>"

# Extract the relevant parts of the message.
mac = data[8:40]
keyId = data[40:56]
encryptionSalt = data[56:64]
authenticationSalt = data[64:72]
ciphertext = data[72:]

# The payload is everything that was signed using the MAC.
payload = data[40:]

# Derive the keys.
encryptionKey = pbkdf2.PBKDF2(password, encryptionSalt, 1000, "SHA1").read(32)
authenticationKey = pbkdf2.PBKDF2(password, authenticationSalt, 1000, "SHA1").read(32)

# Verify with HMAC
valid = verify(authenticationKey, payload, mac)
if not valid:
    sys.exit("Not valid! Message modified!")

print(uuid.UUID(bytes_le=keyId))
print(decrypt(encryptionKey, ciphertext))
```

12.1.3 Node.js

```
const crypto = require('crypto');

/**
 *
 * @param {Buffer} key
 * @param {Buffer} ciphertext
 * @returns A string containing the plaintext.
 */
function decrypt(key, ciphertext)
{
    // The first 16 bytes on the input are the IV.
    // This IV is randomly generated by the sender for each message.
```



```
// We need to separate the IV from the message for proper decoding.
const iv = ciphertext.slice(0, 16);

// The rest is the payload, the actual encrypted message.
const payload = ciphertext.slice(16, ciphertext.length);

var decipher = crypto.createDecipheriv('aes-256-cbc', key, iv);
decipher.setAutoPadding(true); // AutoPadding is by default true. Re-
setting on this example just to avoid confusion.

var plaintext = decipher.update(payload, null, 'utf-
8'); // No inputEncoding is needed because payload is a buffer (param would be ig
nored).
plaintext += decipher.final('utf-8');

return plaintext;
}

/**
 *
 * @param {Buffer} key
 * @param {Buffer} data
 * @returns {Buffer} A buffer containing the hash.
 */
function hash(key, data) {
    const hasher = crypto.createHmac("sha256", key);

    return hasher.update(data).digest();
}

const data = '<Encrypted result goes in here>';
const secret = '<Passphrase goes in here>';

const dataAsBytes = Buffer.from(data, 'base64');

const mac = dataAsBytes.slice(8, 40);
const key = dataAsBytes.slice(40, 56);
const encryptionSalt = dataAsBytes.slice(56, 64);
const authenticationSalt = dataAsBytes.slice(64, 72);
const ciphertext = dataAsBytes.slice(72, dataAsBytes.length);

// The payload is everything that was signed using the MAC.
const payload = Buffer.concat([key, encryptionSalt, authenticationSalt, ciphertext]);
```

```
// Derive the encryption and authentication keys from the secret using PBKDF2.
const encryptionKey = crypto.pbkdf2Sync(secret, encryptionSalt, 1000, 32, 'sha1')
;
const authenticationKey = crypto.pbkdf2Sync(secret, authenticationSalt, 1000, 32,
'sha1');

// After the message has been separated into pieces, verify its integrity by recalculating the hash and comparing it to the provided MAC.
// This allows to ensure that the message has not been modified by anyone in the middle, as if it was, validation would fail.
const verificationHash = hash(authenticationKey, payload);
if (Buffer.compare(verificationHash, mac) !== 0) {
  throw new Error("Could not authenticate the message!");
}

console.log(decrypt(encryptionKey, ciphertext));
```

13 Versioning






The widget is being continuously improved, with new features being added. Some of these changes are small, internal changes; but other changes are bigger and can affect the external structure of the widget, such as the communication with the integrating system, or the presentation and styling. To avoid causing problems in existing integration, the widget is delivered in incremental versions, that you can choose to update at your own pace. With this approach, once you successfully integrate against a specific version, you can be sure that it will continue to work as expected, while at the same time allowing us to keep delivering improvements.

The files distributed by Verisk contain a version number on their path. This applies both to the JavaScript file, and the CSS file. When you receive your integration information, you will receive the latest version, and it is recommended that you always use the latest on a new integration, as it contains more features and improvements than the versions before. After your integration, or even during it, Verisk may release new versions of the widget. You can then check the changelog or contact us to be informed of which improvements have been made and, if you are interested in using them, you simply need to point your files to a new version. For example, if you are using blackbox-3.0.15, and want to update to version 3.0.16, you just need to change the path to the file according to our documentation.

Because a new version is created when there are breaking changes, it is very important that you first verify on your development or test environment how the changes affect your integration. It is also important to be consistent between your environments, to avoid testing on one version and going live with another. The changelog will provide additional information about what has changed, and how to upgrade, when applicable. For the most part, we are committing to provide the easiest upgrade path, so the behavior between versions will be very similar, and only very minor changes on the integration side are needed (if any). For the most part, the only changes you will need to apply are cosmetic, if you have customized the widget presentation and styling.

14 Supported Browsers

The RRT3 Widget supports the most popular web browsers currently available. You can find the list below.

Browser	Version	Operating System	Supported
Microsoft Edge 	Latest	MacOS, Windows, Linux	Yes
Google Chrome 	Latest	MacOS, Windows, Linux	Yes
Mozilla Firefox 	Latest	MacOS, Windows, Linux	Yes
Safari 	Latest	MacOS	Yes
Internet Explorer 	11.x	Windows	Yes

15 Changelog

Version 3.0.19

Released on 2023-04-17

Changes

Changed

- Improved visual contrast of UI elements for compliancy with WCAG 2.1 AA
- Improved screen reader integration for compliancy with WCAG 2.1 AA
- Improvements to height and weight controls

Notes for upgrading

HTML DOM structure and CSS styling has been modified to incorporate the new changes. If your integration uses custom styling, you will need to adapt your CSS.

Version 3.0.18

Released on 2022-03-01

Changes

Added

- Added new customer event system to retrieve results, give control back to client and handle errors on the specified workflows.
- Added Save/Resume screenings feature.
- Added Multiple Language Output feature when completing a screening and returning the results.

Notes for upgrading

No actions needed to update to the new version if the version used is 3.0.17. In case the version used is older than 3.0.17, please refer to the changelog from v3.0.17.

Version 3.0.17

Released on 2021-10-06

Changes

Changed

- Improved responsiveness in mobile devices.

Notes for upgrading

HTML DOM structure and CSS styling has been modified to incorporate the new changes. If your integration uses custom styling, you will need to adapt your CSS.

Version 3.0.16

Released on 2021-02-19

Changes

Added

- Compatibility with the new Automatic Policy Decline feature.

Changed

- Improved accessibility to increase compliancy with WCAG 2.1 AA.
- Improved condition search functionality based on the user language.
- Updated design and content of popup messages to improve clarity.

Fixed

- Small issues when selecting multiple declarable conditions.

Notes for upgrading

HTML DOM structure and CSS styling has been modified to incorporate the new changes. If your integration uses custom styling, you will need to adapt your CSS.