Automation of Document Downloads to Other Systems

Contents

Overview

Using the Whitespace's API and Service Bus Queue systems, it is straight forward to automate the retrieval of documents as they are generated in the platform for storage in your own storage system. This high-level guide provides you with the resources needed to fully implement your document extraction process. (See the document SignedLinesDMSFlow.pdf for an example of a signed line workflow.)

Extracting Documents for Storage in a Storage System

Process

There are two steps to the integration process.

The first step is to connect to one of Whitespace's Azure Service Bus Queues. These inform a queue consumer process about 'activities' generated in the Platform, and are triggered by core user actions and database updates. Each message in the Queue contains a JSON activity document that describes the action that triggered its creation, and contains all the information needed to retrieve all associated documents and data via the API.

Queue consumer services connect to Queues using an Azure Service Bus Shared Access Authorisation URI. This authorisation URI provides 'listen'-only permission to one of the customer's private Azure Service

Whitespace Software Limited – Registered in England & Wales Company No. 02020348
Registered Address – 26th Floor, 22 Bishopsgate, Bishopsgate, London, EC2N 4BQ, United Kingdom

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Whitespace Software Limited. Whitespace is part of Verisk.

Bus queues. Queues are confidential, and are generated for each of a customer's teams. Connection URIs and Queue names are available from Whitespace support on request.

The second step is to call the Whitespace API in response to the activity, using the data contained in the activity document. This allows the retrieval of any or all documents or data required. Your queue consumer will need to be tailored to perform the specific tasks you require in response to the activity types concerned. For example, when an activity document specifies that a contract has been signed, the queue consumer downloads a PDF export of the signed contract, which is then passed to the client's storage system, perhaps via email or Sharepoint.

Message Queue

Messages placed on the Queue are given a title with the format **`"{date} {time} Received message: {activityCode} {activityDocID}"`**. The following image shows an example list of Queue message titles.

> Please note that we include an example javascript Queue consumer process, a detailed explanation of our activity documents, and a full list of activity types in our documentation (see Integrating With Whitespace Via Queues.pdf). Similarly, deeper information on service bus authorisation can be found here: https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-sas

Once the consumer process has identified an activity it should respond to, data and documents are retrieved by making the appropriate calls to the Whitespace API. Calls to the API are validated by a bearer token (JSON Web Token). This can be either a renewable token for the Sandbox test environment (see: Getting a Renewable Token), or a service token (see: Getting a Service Token) for the live system. Service tokens have enhanced security features, including a configurable expiry date and limits on the IP addresses from which API calls can be made.

Key API Calls

When access to the token and connection to a Queue have been established, the most important endpoints used for the process of downloading a contract document are as follows.

GET /api/risks/{activityDocID}

This API call takes the {activityDocID} specified in the Queue message header and returns the full activity document containing the details of the action that triggered that message. The activity document will always be a JSON document containing the **`"type": "RWActivity"`** key-value pair, as in the example below. Note also that the **`"actvity": "Quoted"`** key-value pair identifies the message as having been created by an underwriter providing a quote in response to a broker's request.

```
{
  "_id": "ICF9F146EB-70B8-48AD-98FB-5C754419D9C3::MUB70853CF-3221-4FD4-8A30-12B05EAD2EA5::UQ3::ACTI::00B33825-6F0F-4DBE-844D-F84833F07AE5",
  "activity": "Quoted",
  "channels": [
    "blackpool_MARINE",
    "whitesails_ALLRISKS"
  ],
  "hardcodedActivity": "Quoted",
  "parentDocID": "ICF9F146EB-70B8-48AD-98FB-5C754419D9C3::MUB70853CF-3221-4FD4-8A30-12B05EAD2EA5::UQ3",
  "type": "RWActivity",
  "userID": "MUB70853CF-3221-4FD4-8A30-12B05EAD2EA5"
}
```


This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Whitespace Software Limited. Whitespace is part of Verisk.

For most uses, the most important key is `"parentDocID"`. This identifies the specific document associated with the activity report. The value of parentDocID will always begin with a 38-character alphanumeric string, the {rootID}, uniquely identifying the root contract which the document belongs to. This is optionally followed by one or more extra strings, each separated by a double colon delimiter. These identify the specific document, such as the firm order contract, or a particular request to quote.

Another important key is `"userID"`, which holds the unique Whitespace identifier for the user whose action triggered the activity report. Where the `"owner"` key is present, that identifies the Whitespace user who is primarily responsible for handling the contract at your organisation – it is not included in the example above. Both of these keys can be used to obtain real names, emails, team memberships, and so on.

In the example above, `"parentDocID"` consists of the {rootID} of the contract, delimiters, and then a reference whose initial `MU` indicates that it is a Whitespace user's ID number. In this instance, it is the {userID} of the underwriter who sent the quote, matching the `"userID"` key. Then there is a second delimiter, and the final `UQ3` string shows that this document is the third quote provided by underwriters for the contract.

POST /export/pdf/{parentDocID}

> Please note that this call does not begin with /api, because document export is handled by different resources. It is otherwise formed in the same way as a call to the API resources, including beginning with the environment URL. For details, see the Getting Started with the Whitespace API document.

The call takes the {parentDocID} retrieved from the activity document and a small JSON payload, and returns a zip file containing a PDF or DOCX export of the contract document that is associated with the activity document. The JSON payload required for this call contains the option settings to be used when converting the contract. It is detailed on the Swagger microsite at Whitespace API on Swagger: Export.

GET /api/attachments/{rootID}

The {rootID} of a contract is the unique identifier that refers to the overall contract rather than any specific instance of it. As discussed under the GET /api/risks/{activityDocID} call, all documents associated with a contract have a {parentDocID} that begins with its {rootID} and is then followed by two colons and various other descriptors which identify the document precisely. So retrieving the {rootID} of a contract is as simple as isolating the first 38 characters of {parentDocID}.

Since attachments are associated with the contract as a whole rather than any given instance of it, the call to retrieve attachments requires that you pass it {rootID}, and returns a list of the contract's attachments. Each attachment in the list has a 38-digit {identifier} value beginning with `"A-"` that is paired with its `"identifier"` key.

GET /api/attachments/{rootID}/{identifier}

This simple call takes an individual attachment {identifier} and returns a copy of the attachment.

GET /api/owner/{rootID}

A contract's owner at your organisation is the user who is primarily responsible for handling the contract on the Whitespace platform. Sometimes the `"owner"` key is included in the activity document, with a {userID} value.

In cases where it is not included, it may be retrieved using this simple call. Call the contract's {rootID} with the endpoint, and it returns a short JSON array with various items of information about the contract. The `"ownerUserID"` key in this array gives the {UserID} value for the contract owner at your organisation.

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Whitespace Software Limited. Whitespace is part of Verisk.

GET /api/shared/corporate

This call returns a full JSON list of users, grouped by the organisation that they belong to. This list is the primary method for retrieving the personal information associated with a Whitespace {userID}.

Please note that to ensure permanent correct monitoring and maintenance of contracts, users cannot be deleted from this list, only set to inactive. As such, archiving older versions of the list is unnecessary. The list should be retrieved and stored for reference on your system, and updated whenever you fail to locate a valid {UserID} within it, as that indicates the user is more recent than the last list update.

To identify a specific individual, search the list for a **`"uniqueID"`** key whose value matches the {userID}. This will be part of an array held in the **`"members"`** object. The rest of the array provides the user's identifying information, including **`"name"`**, **`"email"`**, and **`"teams"`**. The **`"companyId"`** key, found at the in the same array as the **`"members"`** object, gives the name of the user's organisation.

A simple example **`"members"`** array is shown below.


```
"members": [
  {
    "uniqueID": "MU5A3310B4-B02E-4C27-95B4-7328C2585F27",
    "name": "Milind Kamu",
    "teams": [
      "AVIATION",
      "MARINE"
    ],
    "currentState": "Live",
    "email": "broker.milind@wspt.co.uk"
  },
```


Storage

Once contract documents and attachments have been retrieved by your system, they will need to be passed into your document management system. Precise details of this process will vary according to the details of your storage system. Common options include email transmission to the storage system, or uploading to Sharepoint.

When your consumer process is emailing documents to a storage system, it will need:

- Origination email address,
- Destination email address,
- A title, possibly containing activity information,
- An email body, again possibly continaing activity information,
- The documents to be attached to the email for storage, and
- Optionally, some cc: or bcc: recipients for the documents.

When uploading a document to Sharepoint, your consumer process will need:

- URL for the Sharepoint folder to upload to,
- A ClientID / username for Sharepoint, and
- A Client Secret / password for Sharepoint.

Other options for lodging documents in the storage system will follow similar logic.

Frequently Asked Questions

1. Is Whitespace integration one-way communication only?
   - Whitespace Queues are one-way, providing JSON documents containing details of activities generated on the Platform.
   - The Whitespace API accepts both GET and POST calls. Between the two, almost every Platform function is fully replicable. POST actions can update the database and generate their own Queue

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Whitespace Software Limited. Whitespace is part of Verisk.

Activity documents. Our core API specification is at https://swagger.whitespace.co.uk although some other, less commonly-used calls are also implemented.

2. Do we need to define the format for messages between the two systems?
    - Whitespace's Queues are made up of JSON activity documents that have a core format for each activity type and include all relevant data. The customer's message consumer process retrieves these activity documents, and then triggers the API calls that are needed to respond to those activities.
3. What is the expected average response time for implementing this service?
    - The Queue consumer is a fully customer-side process. It has to be developed in-house to implement your required needs and responses, and then deployed as normal on your infrastructure, so the time required would be similar for your other operations of similar scale.
    - Messages are generated and added to the queue almost instantaneously, so once your consumer is in place, response time is entirely down to the performance of your consumer process.
4. Can Whitespace provide detailed technical assistance with integration?
    - Ad-hoc technical assistance with specific issues is always available. Technical assistance beyond this level is also available, but will incur T&M charges, either from Whitespace or our integration partners. Whitespace T&M charges are detailed within your Platform Agreement. For more information on this subject, please contact your Whitespace account manager.

Useful Links

· https://apidocs.whitespace.co.uk/ - API documentation (see, specifically, 'Getting Started with the Whitespace API', 'Getting a Renewable Token for API Usage', and 'Integrating with Whitespace via Queues')
· https://swagger.whitespace.co.uk/ - API specification and test harness
· https://support.whitespace.co.uk/home/ - a range of articles, ranging from technical to platform use
· https://support.whitespace.co.uk/kb/ - searchable Knowledgebase
· https://www.whitespace.co.uk/user-guides - more general support (user guides)
· https://www.whitespace.co.uk/support/ - our main support landing page

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Whitespace Software Limited. Whitespace is part of Verisk.