WHITESPACE

INTEGRATING WITH WHITESPACE VIA QUEUES

Updated 7th March 2025, Whitespace Platform v3.1

Contents

WHITESPACE

Overview

Whitespace now offers full, seamless integration to its systems via the Microsoft Azure Service Bus Queues interface. This article offers a simple overview of that process from a development perspective.

Whenever a user performs a significant action on the Whitespace platform, the system generates a JSON activity document summarising that action. These summaries contain all the information necessary for an authenticated process to precisely understand the action. The activity document is then placed on the client's queue, timestamped and ready to be retrieved by a consumer process.

There are several advantages inherent to the Queues interface:

- Language-agnostic, for maximum flexibility.
- Served in strict order, so processing accurately reflects the flow of actions.
- No hard requirement for synchronous processing, simplifying load-balancing.
- Settlement systems help when ensuring correct and unique processing.
- Loose integration ensures upgrades and system changes never affect the data flow.

Within the Whitespace system, every significant state change is assigned a unique plain-text identifier, an activity string. This precisely identifies the action that spawned the activity document, and is contained within it. Along with the other information within the document, this allows the message consumer to parse the action and, through API calls, gain a complete understanding of it.

A fuller non-technical overview of Whitespace's use of Queues can be found at https://apidocs.whitespace.co.uk/Queues_and_the_Whitespace_Platform.pdf.

For access to your message queue, contact the Whitespace Integration Team or email support@whitespace.co.uk. Whitespace will provide you with the queue's name, key, and connection string. These will permit you to interrogate the queue.

Receiving Queue Messages

The versatility of the Azure environment provides the integrator with significant flexibility when preparing a message consumer process. Typically, the core process flow for a message consumer is to poll the queue for the oldest message, and when one is present, retrieve the data it contains, process this according to the integrator's desired logic, and confirm to the queue that the message was correctly received. On confirmation of delivery, the message is deleted from the queue, and the next oldest rises to the top.

Consumer processes can stream messages, loop message reception, open sessions with the queue, use Azure APIs to trigger push functionality, and otherwise be configured according to the client's precise needs. Multiple consumers can run in parallel, utilizing the time-stamping of the messages to ensure smooth processing. Queues contain functionality to help avoid duplicate messages, and delivery modes ensure no message is ever missed. 'Read Exactly Once' functionality can be achieved with some simple trapping of received message ID number.

Note that Microsoft offers library samples of Azure Service Bus handlers for .NET, Java, Python, JavaScript, and TypeScript at the https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-queues-topics-subscriptions web page. General SDKs are available for Go, C++, C, Android, and iOs.

A simple JavaScript program to read a queue message and look for some basic activities is shown below.

```
const yargs = require("yargs");
const fs = require('fs');
const { delay, ServiceBusClient, ServiceBusMessage } = require("@azure/service-bus");

const options = yargs
 .usage("Usage: node receive.js -s settings.json")
 .option("s", { alias: "settings", describe: "JSON Settings File", type: "string", demandOption: true })
 .option("v", { alias: "verbose", describe: "Verbose", type: "boolean", demandOption: false })
 .argv;

async function main() {
        console.log( `${myTimeStamp()} Loading ${options.settings}`)
        fs.readFile(options.settings, (err, data) => {
                if(err){
                        console.log(`Error reading ${options.settings}`);
                } else {
                        const obj = JSON.parse(data);
                        if( obj.queueName && obj.connectionString ) {
                                readQueue(obj.queueName, obj.connectionString);
                        } else {
                                console.log( "Error: expected queueName and connectionString");
                                console.log( obj );
                        }
                }
        })
}

async function readQueue( queueName, connectionString ) {
        console.log( `${myTimeStamp()} Reading messages from ${queueName}`);

        // create a Service Bus client using the connection string to the Service Bus namespace
        const sbClient = new ServiceBusClient(connectionString);

        // createReceiver() can also be used to create a receiver for a subscription.
        const receiver = sbClient.createReceiver(queueName);

        // function to handle messages
        const myMessageHandler = async (messageReceived) => {
                const type = messageReceived.body.type || "No type";
                let message = messageReceived.body.hardcodedActivity || "No hardcodedActivity";
                if( type == 'RWComment' ) {
                        message = messageReceived.body.comment || "No comment";
                }
                console.log(`${myTimeStamp()} Received ${type} message: ${message} ${messageReceived.body._id}` );
                if( options.verbose ) {
                        console.log( messageReceived.body );
                }
                if( type == 'RWComment' ) {
                        handleComment( messageReceived.body );
                        return;
                }
                if( messageReceived.body.hardcodedActivity == "Quoted" || messageReceived.body.hardcodedActivity == "Quoted with Quote Details") {
                        handleQuoted( messageReceived.body );
                }
                if( messageReceived.body.hardcodedActivity == "Signed Lines" ) {
                        handleSigned( messageReceived.body );
                }
        };

        const handleComment = async( comment ) => {
                console.log( `Comment ${comment.user} said: ${comment.comment}`)
        }

        const handleQuoted = async( activity ) => {
                console.log( `Handling ${activity.hardcodedActivity} in document ${activity._id}`);
                console.log( `Will read the document using GET /api/v22.04/data/${activity.parentDocID}`);
                console.log( `and from that extract e.g. premium to be considered by colleagues`);
        };
        const handleSigned = async( activity ) => {
                console.log( `Handling ${activity.hardcodedActivity} in document ${activity._id}`);
                console.log( `Will read the Signed PDF using GET /export/pdf/$RiskID where $RiskID is the ::SIGNED version`);
                console.log( `and store that in a document repository`);
        };

        // function to handle any errors
        const myErrorHandler = async (error) => {
                console.log(error);
        };

        // subscribe and specify the message and error handlers
        receiver.subscribe({
                processMessage: myMessageHandler,
                processError: myErrorHandler
        });
}

function myTimeStamp() {
        const dateStamp = new Date().toLocaleDateString();
        const timeStamp = new Date().toLocaleTimeString();
        return `${dateStamp} ${timeStamp}`;
}

main().catch((err) => {
        console.log("Error occurred: ", err);
        process.exit(1);
});
```

By default, only RWActivity documents (and, if activated by the user, RWCustomActivity documents) are placed on the queue. RWComment documents representing chat messages can be added as well if the integration requires them. In the example above, note that line 39 retrieves the **type** value from the message. RWComment files are very simple logs of text comments involving the client's users – the ones associated with that queue, anyway. Clients may use more than one distinct queue.

All other significant system events are handled using documents of the type RWActivity. The **body.hardcodedActivity** key references the activity string of the action that generated that RWActivity document. In the example above, lines 52 to 57 demonstrate indentifying specific activity strings, which are then passed to the relevant dummy handler process in lines 64 to 73. Here, 'Quoted' and 'Quoted with Quote Details' are both passed to the same handler, because they are very similar in structure and, for most use cases, function. RWCustomActivity documents are very similar in structure, except that they provide queue entries for custom actions taken by a user, such as triggering an integration-side consumer process to call a third party rating service.

A plain text version of this example receive.js file is attached as Appendix A.

Parsing Queue Messages

Since RWActivity documents are generated by all actions, they are quite variable in specific form. A reasonably detailed example is attached as Appendix B, but a heavily truncated version is shown below.


```
{
  "_id": "ICBC2647B7-778F-49D3-84AC-316F8C297ACB::FO::ACTI::54309426-C8CF-4027-8EFE-060CE31EF902",
  "_rev": "1-0b1572b355539ebac561d37b8797ab7d",
  "activity": "Requested a Line",
  "apnsData": {
    "channels": [
      "palermo_ALL"
     ],
    "subscriptionRootID": "ICBC2647B7-778F-49D3-84AC-316F8C297ACB",
    "subtitle": "AJC Broking Ltd",
    "title": "Requested a Line"
    },
  "channels": [
    "palermo_ALL",
    "ajc_PROPERTYAFRICA"
    ],
  "createdAt": "2022-05-25 11:16:46",
  "hardcodedActivity": "Requested a Line",
  "linkedArchiveID": "ICBC2647B7-778F-49D3-84AC-316F8C297ACB::FO::ARCH::27287664-6DFE-4F1F-AA62-1FEBD2C40173",
  "parentDocID": "ICBC2647B7-778F-49D3-84AC-316F8C297ACB::FO",
  "type": "RWActivity",
  "userID": "MUD38EC011-780A-42D3-94DA-FD9063F5DAF9"
}
```


In the above excerpt, the most salient keys are _id, createdAt, hardcodedActivity, parentDocID, and userID. The value of _id is the unique message reference, with createdAt identifying the time, and userID indicating which of the client's users the action is attached to.

The root risk that the activity relates to is identified by the first 38 characters of parentDocID and _id, as well as the linkedArchiveID and subscriptionRootID – all root risks in Whitespace are uniquely identified by a string consisting of the letters IC followed by 32 characters separated by dashes. All other associated documents, including specific instances of the contract, are identified by the data following the root ID. In this example, the :: spacer followed by the code FO specifies that the contract is at the 'Firm Order' stage. Elsewhere in the excerpt, ::ACTI indicates an activity document, and ::ARCH an archive document.

The value of the hardcodedActivity key provides the identifier for the action that spawned the RWArchive document and placed it on the queue. A full list of activity strings is attached as Appendix C. In this example, "Requested a Line" specifies that a broker placed a firm order for a written line with a lead underwriter. We can see from the apnsData object literal, associated with text and email message generation, that a notification was sent for the attention of Palermo's 'All staff' team channel.

Passing the appropriate document IDs to the Whitespace API allows the consumer to obtain every possible detail relating to this request. Calling GET /risks/root/{rootID} will return a full list of all documents under that

root risk, for example, while GET /documents/getLineGuidanceForRoot/{rootID} will show the percentages that the broker has suggested for the written lines, and GET /activities/{riskID}/full will return a complete timestamped history of everything that has happened to that document. The {riskID} token can accept full document identifiers, while the {rootID} token only accepts the risk's root identifier.

For further details of the Whitespace API system, see http://apidocs.whitespace.co.uk and http://swagger.whitespace.co.uk.

APPENDIX A

Plain Text of Receive.js

```
const yargs = require("yargs");
const fs = require('fs');
const { delay, ServiceBusClient, ServiceBusMessage } = require("@azure/service-bus");

const options = yargs
 .usage("Usage: node receive.js -s settings.json")
 .option("s", { alias: "settings", describe: "JSON Settings File", type: "string", demandOption: true })
 .option("v", { alias: "verbose", describe: "Verbose", type: "boolean", demandOption: false })
 .argv;

async function main() {
        console.log( `${myTimeStamp()} Loading ${options.settings}`)
        fs.readFile(options.settings, (err, data) => {
                if(err){
                        console.log(`Error reading ${options.settings}`);
                } else {
                        const obj = JSON.parse(data);
                        if( obj.queueName && obj.connectionString ) {
                                readQueue(obj.queueName, obj.connectionString);
                        } else {
                                console.log( "Error: expected queueName and connectionString");
                                console.log( obj );
                        }
                }
        })
}

async function readQueue( queueName, connectionString ) {
        console.log( `${myTimeStamp()} Reading messages from ${queueName}`);

        // create a Service Bus client using the connection string to the Service Bus namespace
        const sbClient = new ServiceBusClient(connectionString);

        // createReceiver() can also be used to create a receiver for a subscription.
        const receiver = sbClient.createReceiver(queueName);

        // function to handle messages
        const myMessageHandler = async (messageReceived) => {
                const type = messageReceived.body.type || "No type";
                let message = messageReceived.body.hardcodedActivity || "No hardcodedActivity";
                if( type == 'RWComment' ) {
                        message = messageReceived.body.comment || "No comment";
                }
                console.log(`${myTimeStamp()} Received ${type} message: ${message}
${messageReceived.body._id}` );
                if( options.verbose ) {
                        console.log( messageReceived.body );
                }
                if( type == 'RWComment' ) {
                        handleComment( messageReceived.body );
                        return;
                }
                if( messageReceived.body.hardcodedActivity == "Quoted" ||
messageReceived.body.hardcodedActivity == "Quoted with Quote Details") {
                        handleQuoted( messageReceived.body );
                }
                if( messageReceived.body.hardcodedActivity == "Signed Lines" ) {
```

```
                        handleSigned( messageReceived.body );
                }
        };

        const handleComment = async( comment ) => {
                console.log( `Comment ${comment.user} said: ${comment.comment}`)
        }

        const handleQuoted = async( activity ) => {
                console.log( `Handling ${activity.hardcodedActivity} in document ${activity._id}`);
                console.log( `Will read the document using GET /api/v22.04/data/${activity.parentDocID}`);
                console.log( `and from that extract e.g. premium to be considered by colleagues`);
        };
        const handleSigned = async( activity ) => {
                console.log( `Handling ${activity.hardcodedActivity} in document ${activity._id}`);
                console.log( `Will read the Signed PDF using GET /export/pdf/$RiskID where $RiskID is`);
                console.log( `the ::SIGNED version and store that in a document repository`);
        };

        // function to handle any errors
        const myErrorHandler = async (error) => {
                console.log(error);
        };

        // subscribe and specify the message and error handlers
        receiver.subscribe({
                processMessage: myMessageHandler,
                processError: myErrorHandler
        });
}

function myTimeStamp() {
        const dateStamp = new Date().toLocaleDateString();
        const timeStamp = new Date().toLocaleTimeString();
        return `${dateStamp} ${timeStamp}`;
}

main().catch((err) => {
        console.log("Error occurred: ", err);
        process.exit(1);
});
```

APPENDIX B

Sample RWActivity Document


```
{
  "RWLineGuidanceSet": {
    "_id": "ICBC2647B7-778F-49D3-84AC-316F8C297ACB::FO::PALERMO::ALL::LGUS",
    "_rev": "1-0c9aef60c2398520d0d1ff3caa335c6c",
    "associatedPlacingID": "ICBC2647B7-778F-49D3-84AC-316F8C297ACB::FO",
    "channels": [
      "ajc_PROPERTYAFRICA",
      "palermo_ALL"
      ],
    "contents": [
      {
        "conditionsEnabled": true,
        "maxString": "100",
        "minString": "100",
        "sectionIdentifiers": [],
        "suggestedStamps": []
        }
      ],
    "createdAt": "2022-05-25 11:16:46",
    "provenance": {
      "dataHash": "d02b68575fb8dc19803ba819456dc1e75f959b05e3c8ba85cdddde2466dff9c4",
      "provHash": "dde8b6c89592df946a1ec6161c8cdb2e132017d6f7af395520c89610746d34e0",
      "system": "LaVAPI",
      "userID": "MUD38EC011-780A-42D3-94DA-FD9063F5DAF9",
      "version": "2022-05-24@19:08:46",
      "writtenAt": "2022-05-25 11:16:46"
      },
    "sections": [],
    "type": "RWLineGuidanceSet",
    "updatedAt": "2022-05-25 11:16:46",
    "userID": "MUD38EC011-780A-42D3-94DA-FD9063F5DAF9"
    },
  "_id": "ICBC2647B7-778F-49D3-84AC-316F8C297ACB::FO::ACTI::54309426-C8CF-4027-8EFE-060CE31EF902",
  "_rev": "1-0b1572b355539ebac561d37b8797ab7d",
  "activity": "Requested a Line",
  "apnsData": {
    "channels": [
      "palermo_ALL"
      ],
    "data": {
      "docId": "ICBC2647B7-778F-49D3-84AC-316F8C297ACB"
      },
    "dataForEmail": {
      "riskInformation": [
        "Hendrix Estates Inc",
        "USD 100,000,000",
        "B0999JC2505221216"
        ],
      "userID": "MUD38EC011-780A-42D3-94DA-FD9063F5DAF9"
      },
    "subscriptionRootID": "ICBC2647B7-778F-49D3-84AC-316F8C297ACB",
    "subtitle": "AJC Broking Ltd",
    "title": "Requested a Line"
    },
  "channels": [
    "palermo_ALL",
```

```
    "ajc_PROPERTYAFRICA"
    ],
  "createdAt": "2022-05-25 11:16:46",
  "hardcodedActivity": "Requested a Line",
  "linkedArchiveID": "ICBC2647B7-778F-49D3-84AC-316F8C297ACB::FO::ARCH::27287664-6DFE-4F1F-AA62-1FEBD2C40173",
  "parentDocID": "ICBC2647B7-778F-49D3-84AC-316F8C297ACB::FO",
  "provenance": {
    "dataHash": "d55a5416de924f0f68878781c132e4bdd1b223f7352e2c9308bb70d12b97a884",
    "provHash": "02d66931189e1b72e3d5997c4bf6792d8a23204382f76baa6c09617592ceeefa",
    "system": "LaVAPI",
    "userID": "MUD38EC011-780A-42D3-94DA-FD9063F5DAF9",
    "version": "2022-05-24@19:08:46",
    "writtenAt": "2022-05-25 11:16:46"
    },
  "type": "RWActivity",
  "userID": "MUD38EC011-780A-42D3-94DA-FD9063F5DAF9"
}
```

APPENDIX C

Alphabetical List of Activity Strings

The Activity String identifies the precise action taken by a user to generate the Queues message. Each action is explained in more detail in the list below. All relevant details are specified in or accessible via the activity document forming the main body of the message, including the relevant user/s and the full contract, as detailed above. This list is complete as of the platform version shown at the beginning of this document, and custom user-defined activity strings are also available.

In the following list, the user type associated with each activity string is listed after the string's title in brackets.

- 'Broker' and 'Underwriter' indicate that the activity can only be generated by users from organisations with those respective roles.
- 'Either' indicates that the activity can be generated by both types of user.
- 'Reserved' indicates an activity string that is not currently implemented, but which we envision a future use for.

For a list of the activity strings that should be anticipated at each stage of the contract placing process, please see Appendix D.

A Line Was Removed *(Underwriter)*: The underwriter has removed their agreement of cover from the contract in response to the broker's inability to satisfy one or more of their conditions of that agreement.

A Subjectivity Was Removed *(Underwriter)*: The underwriter has removed one of their conditions of providing insurance from the contract.

Accepted a Quote *(Broker)*: The broker has accepted the underwriter's quoted contract as the basis of a firm order.

Accepted a Quote Request *(Reserved)*: TBC

Accepted Endorsement *(Underwriter)*: The underwriter has agreed an endorsement to a firm order or signed contract.

Accepted on behalf of Buyer *(Reserved)*: TBC

Added an Attachment *(Either)*: The user has attached a document to the contract.

Added an Attachment from a Subjectivity Response *(Broker)*: The broker has attached a document to the contract in response to an underwriter's condition of providing insurance.

Attachment deleted *(Broker)*: The broker has deleted an unshown attachment that they previously added to the contract. Please note that this is a broker-only activity because attachments added by underwriters are immediately shown to the broker, and attachments that have been shown to another party cannot be deleted.

Attachment(s) shown *(Broker)*: The broker has shown one or more contract attachments to one or more underwriters. Please note that this is a broker-only activity because attachments added by underwriters are automatically shown to the broker, and cannot shared within the Whitespace Platform.

Bindable Line Written *(Underwriter)*: The underwritten has provided a commitment to insure a percentage of a bindable quote contract as specified in their line details.

Bindable Quote Not Taken Up *(Broker)*: The broker has marked a received bindable quote as 'Not Taken Up'.

Bindable Quote Shown *(Broker)*: The broker has shown a bindable quote to one or more underwriters.

Bound notify party *(Broker)*: The broker has bound all notify and non-notify parties on the facility/line slip agreement to the declaration or endorsement, following the lead underwriter's acceptance of the same.

Broker Attempted To Satisfy A Subjectivity *(Broker)*: The broker has written a reply in response to an underwriter's requirements on the contract.

Broking Partner Request *(Broker)*: The broker has sent a contract as a producing document to another broker for placing.

Change rejected (Conflict) *(Reserved)*: TBC

Changed or Added a Line item *(Either)*: The user has either added or modified a contract heading and/or its associated text. Any alteration under a contract heading which requires clicking 'Save' in the text editor will generate this activity, including tagging defined data and altering formatting. Please note that brokers can only edit unshown draft contracts and unshown firm orders, and underwriters can only edit unshown quote contracts.

Changed or Added Quote Details *(Underwriter)*: The underwriter has filled in or revised some or all of the meta-data in the 'Quote Details' pane, such as 'Quoted Line Percentage' or 'Validity Period'. This pane is inserted above the top of the contract when responding to a quote request, and does not form part of the contract text. Its use is optional.

Cloned a Quote *(Underwriter)*: The underwriter has made a duplicate copy of a quote, with or without 'Quote Details', permitting them to offer alternate terms to a broker.

Cloned Endorsement *(Broker)*: The broker has duplicated an existing endorsement. Please note that cloned endorsements do not have to be applied to the same contract as the original.

Cloned From Declaration *(Broker)*: The broker has duplicated an existing contract declared to a facility/line slip as a new draft contract. Please note that the resulting draft contract will not be automatically set as a declaration, but will include the contract heading necessary for manually identifying a specific facility/line slip's agreement contract.

Cloned From Risk *(Broker)*: The broker has duplicated an existing contract as a new draft contract.

Cloned From Template *(Broker)*: The broker has duplicated an existing contract template as a new draft contract.

Completed Endorsement *(Broker)*: The broker has completed the endorsement following its acceptance by any leader and/or agreement parties and/or the binding of notify and non-notify parties.

Contract Correction Approved *(Underwriter)*: The underwriter has approved a contract correction.

Contract Correction Completed *(Broker)*: The broker has applied a unanimously-agreed correction to the contract, updating it.

Contract Correction Declined *(Underwriter)*: The underwriter has declined a contract correction.

Contract Correction Deleted *(Broker)*: The broker started the contract correction process but deleted the correction document unshown.

Contract Correction Revoked *(Broker)*: The broker has revoked a contract correction, keeping the contract in its uncorrected state. This is the only action available for a contract correction that has been declined by any underwriters, but fully-approved corrections can also be revoked if appropriate.

Contract Correction Shown *(Broker)*: The broker has shown a contract correction document to one or more underwriters.

Contract Correction Withdrawn *(Broker)*: The broker has withdrawn a shown contract correction before any underwriter provided a response.

Contract Published Succesfully *(Reserved)*: TBC

Contract Updated *(Reserved)*: TBC

Created Bindable Quote *(Broker)*: The broker has created a bindable quote from an existing template, draft, or quoted contract.

Created New Draft *(Broker)*: The broker has created a new draft contract, without cloning an existing document, by selecting 'Digitise', 'Full Contract', 'Off-Platform Contract' or 'Outline Contract' from the 'Create New+' option button on the main page.

Created New Endorsement *(Broker)*: The broker has created a new contract endorsement without cloning an existing endorsement.

Created New Placement *(Reserved)*: TBC

Declaration shown to Facility *(Broker)*: The broker has shown a contract to the underwriters participating in a facility/line slip agreement.

Declined Bindable Quote *(Underwriter)*: The underwriter has decided to reject a broker's request write a line on a bindable quote.

Declined Endorsement *(Underwriter)*: The underwriter has decided to reject an endorsement to a contract.

Declined Firm Order *(Underwriter)*: The underwriter has decided to reject a broker's request to provide insurance.

Declined Quote Request *(Underwriter)*: The underwriter has decided to not provide a quote on a contract.

Deleted contract sections *(Either)*: The user has removed an entire contract section from a contract during editing, deleting all the contract headings that fall under that contract section, along with their associated text. Please note that brokers can only edit unshown draft contracts and unshown firm orders and underwriters can only edit unshown quotes.

Draft was Deleted *(Broker)*: The broker has deleted an unshown draft contract.

Endorsement Not Taken Up *(Reserved)*: TBC

Endorsement was Deleted *(Broker)*: The broker has deleted an unshown draft endorsement.

Internal Broker Notes Edited *(Broker)*: The broker has edited the internal notes of a contract.

Internal Review: Contract Approved *(Either)*: A user with the 'Internal Review' status has reviewed and approved a contract as ready to progress on behalf of a team-member.

Internal Review: Contract Rejected *(Either)*: A user with the 'Internal Review' status has reviewed and rejected a contract as not ready to progress on behalf of a team-member.

Internal Review: Request Withdrawn *(Either)*: A contract has been removed from the internal review queue because the internal review request was cancelled.

Internal Review: Requested *(Either)*: A contract has been flagged for internal review. Please note that the contract cannot be progressed further until it has been approved. This request can be triggered voluntarily by the user requesting the review, or automatically, when the user's team has been set for compulsory internal review of contracts in the platform's administration portal.

Internal Review: Self-Approved *(Both)*: The user has self-approved a contract instance of theirs that was flagged for internal review.

Library IDs Associated with Line Items *(Broker)*: The broker has imported a template into the Contract Library, linking Library Line Items with their corresponding contract headings in the template.

Line Conditions/Subjectivities were updated *(Underwriter)*: The underwriter has specified or revised their list of required conditions for accepting the contract.

Line Percentage Was Revised *(Underwriter)*: The underwriter has revised their written line percentage on the contract in response to the broker's proposal.

Line Reference Was Revised *(Underwriter)*: The underwriter has revised their risk code/s, reference/s, or other details on the contract.

Line Removal Accepted (*Underwriter*): The underwriter has accepted a broker's request to remove their line from a firm order.

Line Removal Rejected (*Underwriter*): The underwriter has rejected a broker's request to remove their line from a firm order.

Line Removal Request (*Broker*): The broker has requested authorisation to remove an underwriter's line from a firm order.

Line Removed (*Broker*): The broker has removed an off-platform line from a firm order contract or a (non-) notification line from a firm order declaration.

Line Written *(Underwriter)*: The underwriter has committed to insure a percentage of the contract as specified in their written line details.

Marked as Firm Order *(Broker)*: The broker has committed to using the specific draft contract or underwriter's quote as the contract of insurance, and that contract has been advanced to the 'Firm Order' stage.

Marked Facility Active *(Broker)*: The broker has marked a facility agreement as 'Active'.

Marked Facility Expired *(Broker)*: The broker has marked a facility agreement as 'Expired'.

Marked Facility Inactive *(Broker)*: The broker has marked an off-platform facility agreement as 'Inactive'

Marketed Risk *(Broker)*: The broker has shared a draft contract with 'Market Only' status, permitting the recipients to read the contract but not to quote on it.

Mid-Term Participant Change Completed *(Broker)*: The broker has completed a mid-term participant change endorsement.

Offered Quote *(Reserved)*: TBC

Off-Platform Bindable Line Written *(Broker)*: The broker has detailed a commitment agreed with an off-platform underwriter to insure a percentage of a bindable quote contract as specified in their line details.

Off-Platform line was adjusted *(Broker)*: The broker has adjusted the details of a written line agreed with an underwriter who does not yet have an account on the Whitespace Platform.

Off-Platform Line Written *(Broker)*: The broker has recorded the details of a written line agreed with an underwriter who does not yet have an account on the Whitespace Platform.

Placing Updated *(Reserved)*: TBC

Proposed a new Written Line Percentage *(Broker)*: The broker has suggested that an underwriter take on a different percentage of a contract to that which they have already written.

Questionnaire Answers Updated *(Broker)*: The Template Manager broker has saved default answers to one or more questions in a questionnaire.

Questionnaire Removed *(Broker)*: The Template Manager broker has completely removed the questionnaire section from a template.

Questionnaire Structure Updated *(Broker)*: The Template Manager broker has added or removed one or more questions from a template's questionnaire.

Quote Being Prepared was Deleted *(Underwriter)*: The underwriter deleted a previously-saved quote that had not yet been shown to a broker.

Quote in Preparation *(Underwriter)*: The underwriter has opened a broker's request for quote.

Quote Not Taken Up *(Broker)*: The broker has marked a received quote as 'Not Taken Up'.

Quote Requested *(Broker)*: The brokers has sent a draft contract to an underwriter, asking for them to quote a percentage of the contract that they are prepared to insure.

Quote was forwarded as a Marketed risk *(Broker)*: The broker has shared an underwriter's quote with other underwriters as a 'Market Only' document that cannot be directly responded to.

Quote was forwarded as a Request To Quote *(Broker)*: The broker has shared an underwriter's quote with other underwriters as the basis for them to prepare their own quotes from.

Quoted *(Underwriter)*: The underwriter has sent a quote to a broker without filling in any fields of the optional 'Quote Details' box.

Quoted with Quote Details *(Underwriter)*: The underwriter has sent a quote to a broker having entered some content into one or more of the fields of the optional 'Quote Details' box.

Replaced an Attachment *(Either)*: The user has replaced an older attachment file with a newer one. Note that while the older file no longer shows in the attachment pane, it remains accessible to all parties.

Request Received from Unapproved Broker (*Underwriter*): The underwriter's organisation has the Approved Broker functionality active, and the underwriter has received a request for a quote, firm order, or bindable quote from a broker who is not on the organisation's list of Approved Brokers.

Request to Quote cloned to a Draft *(Broker)*: The broker has duplicated an existing contract already sent as a request for quotation as a new draft contract.

Requested a Line *(Broker)*: The broker has sent a formal request for insurance cover to one or more underwriters.

Re-Signed Line(s) *(Broker)*: The broker has signed the contract again, following an agreed correction to the contract after the initial signing.

Responded to Subjectivity Response *(Broker)*: The broker has marked an underwriter's requirement for providing cover as provisionally fulfilled.

Responded to Unapproved Broker (*Underwriter*): The underwriter's organisation has the Approved Broker functionality active, and the underwriter has responded to a request for a quote, firm order, or bindable quote from a broker who is not on the organisation's list of Approved Brokers.

Returned to Broking Partner *(Broker)*: The broker has returned a contract as a placing document to the producing broker.

Risk Has Been Deleted *(Reserved)*: TBC

Section Assigned to Shown Contract *(Broker)*: The broker has added sections to a previously unsectioned contract that has already been shown at the firm order or Bindable Quote stages.

Sent New Quote Request *(Broker)*: The broker has sent a revised quote request to an underwriter who has already been asked to quote on the contract.

Sent to Broker *(Reserved)*: TBC

Shared Endorsement *(Broker)*: The broker has completed an endorsement on a master facility which has been shared with one or more team/s in their organisation

Shared Facility *(Broker)*: The broker has shared a master facility with another team in their organisation.

Showed Endorsement *(Broker)*: The broker has shown the endorsement to one or more underwriters.

Showed Marketed Risk for Quote *(Broker)*: The broker has asked one or more underwriters to provide a quote on a contract previously shared on the Platform as 'Market Only'.

Showed to Following Market *(Broker)*: Following its acceptance by the lead underwriter, the broker has requested that one or more agreement parties to a facility/line slip provide written lines on a declared contract.

Signed Contract reverted to Firm Order (*Broker*): The broker has reverted a signed contract back to an unsigned firm order contract, retaining its written lines.

Signed Lines *(Broker)*: The broker has signed the contract, finalising it.

Stamp Update Accepted (*Broker*): The broker has accepted an underwriter's stamp update request.

Stamp Update Rejected (*Broker*): The broker has rejected an underwriter's stamp update request.

Stamp Update Submitted (*Underwriter*): The underwriter has submitted a stamp update request to modify the stamp on a line they have put down.

Subjectivities/Line Conditions Accepted *(Broker)*: The broker has accepted the underwriter's requirements for providing cover, and will proceed to fulfil them.

Subjectivities/Line Conditions Rejected *(Broker)*: The broker has rejected the underwriter's requirements for providing cover, and the quote or written line associated with them.

Template Applied *(Broker)*: The broker has merged the defined data from a condensed contract with the text of a detailed template to produce a full contract.

Template Visibility *(Broker)*: The broker has added or removed access for a template to members of another user team within the same Whitespace corporate account. (Templates are always accessible by at least one active user team.)

Vertical Terms Updated *(Broker)*: The broker has added vertical terms to the text falling under a contract heading, or has modified previously added terms.

Viewed *(Reserved)*: TBC

Viewed Attachment *(Either)*: The user has viewed an attachment which they have not previously opened.

Withdrew Bindable Quote *(Broker)*: The broker has withdrawn a request for an underwriter to write a bindable line.

Withdrew Endorsement *(Broker)*: The broker has withdrawn an endorsement to a contract. Note that an endorsement can only be withdrawn before it has been accepted by an underwriter.

Withdrew Firm Order *(Broker)*: The broker has decided to withdraw a previously-offered firm order. Note that a firm order to a specific underwriter can only be withdrawn before their line has been written.

Withdrew Quote *(Underwriter)*: The underwriter has decided to withdraw a quote from consideration. Note that a quote can only be withdrawn until it is taken up as the basis of the broker's firm order.

Withdrew Quote Request *(Broker)*: The broker has decided to withdraw their request for an underwriter to quote on a contract. Note that a quote request can only be withdrawn until the underwriter provides a quote in response.

Wrote No Cover Given Line *(Underwriter)*: The underwriter has written a line that gives no cover.

WHITESPACE

APPENDIX D

Process Phases and Related Activities

Here we describe the phases of system process flow, from the broker's initial preparation of a draft contract through to the adding of an endorsement to a completed or nearly-completed contract. In each case we indicate the activity codes that can be generated, with a brief explanation. For full details of each activity code, see Appendix C.

The specific contract stages recognised by the Whitespace Platform are: *Draft*, *Marketed*, *Request to Quote*, *Quote, Bindable Quote, Firm Order*, *Signed*, and *Endorsement*. Note that contracts can also be flagged as *Facility*, if they are the founding contract of a facility/line slip agreement, and *Declaration*, if they have been shown to the members of a facility/line slip, but these flags are not considered to be process stages.

Phase 1 : Creation

The broker creates a new draft contract

Template Visibility – team access to a specific template has been added or removed

Library IDs Associated with Line Items – A template was imported into the Contract Library and its contract headings linked to the resulting Library Line Items.

Questionnaire Structure Updated – questions have been added to, or removed from, a template's questionnaire.

Questionnaire Answers Updated – default answers have been saved for questions in a questionnaire.

Questionnaire Removed – a questionnaire has been removed from a template.

Cloned From Template – using a template

Created New Draft – using the 'Create New+' button

Cloned From Risk – using an old contract

Request to Quote cloned to a Draft – using a contract at the request to quote stage

Cloned From Declaration – using a contract previously declared to a line slip/facility

Broker Partner Request – draft placing contract sent by producing broker

Template Applied – broker has merged data from one contract with text from a template to produce a full contract

The broker modifies a draft contract

Changed or Added a Line item – contract heading added and/or text edited in draft

Deleted contract sections – section heading/s removed from draft

Vertical Terms Updated – text was verticalised, or vertical terms updated

Draft was Deleted – contract deleted unshown

Internal Broker Notes Edited – broker's internal team notes on a contract revised

The broker modifies the accessibility of a completed facility for members of other teams in their organisation

Shared Facility – original ('master') facility agreement shared with another team

Shared Endorsement – shared master facility agreement successfully endorsed, updating shared version/s

Marked Facility Expired – on-platform (regular) facility agreement flagged as 'Expired'

Marked Facility Inactive – off-platform facility agreement flagged as 'Inactive'

Marked Facility Active – facility agreement flagged as 'Active'

Phase 2 : Attachments

Any party can add attachments to an existing contract at any phase of the process

Added an Attachment – supplementary document attached

Attachment deleted – attached document deleted before being shown to any parties

Attachment(s) shown – attached document shown

Replaced an Attachment – attached document replaced

Viewed Attachment – attached document opened

Phase 3: Quotation

The broker shows the contract to underwriters
Quote Requested – request for quote sent
Request Received from Unapproved Broker – underwriter gets request from broker not in Approved Broker list
Declaration shown to Facility – contract declared to a facility/line slip
Marketed Risk – contract shown but quotes not invited
Created Bindable Quote – contract marked as a bindable quote
Bindable Quote Shown – request for bindable line sent
Section Assigned to Shown Contract – sections have been added to an unsectioned shown bindable contract
Sent New Quote Request – contract reshown for quote
Showed Marketed Risk for Quote – formerly marketed contract now shown for quote
Off-Platform Bindable Line Written – details have been recorded of a bindable deal with an underwriter not in the system
Withdrew Quote Request – contract withdrawn
Withdrew Bindable Quote – bindable contract withdrawn

The underwriter prepares a quote for the broker
Response to Unapproved Broker – underwriter replies to a broker not in Approved Broker system list
Quote in Preparation – quote request opened
Declined Quote Request – quote request refused
Declined Bindable Quote – bindable line request refused
Changed or Added a Line item – contract heading added and/or text edited in quote
Changed or Added Quote Details – optional quote meta-data entered
Deleted contract sections – section heading/s removed from quote
Cloned a Quote – quote duplicated
Quote Being Prepared was Deleted – quote deleted unshown

The underwriter sends a quote to the broker
Quoted – quote returned
Quoted with Quote Details – quote returned with optional meta-data
Bindable Line Written – bindable quote returned with a written line
Withdrew Quote – previously sent quote withdrawn

The broker sends one underwriter's quote to others
Quote was forwarded as a Request To Quote – quote shown to other underwriter/s to quote from
Quote was forwarded as a Marketed risk – quote shown to other underwriters but quotes not invited
Returned to Broking Partner – contract instance returned by placing broker to producing broker

Phase 4: Checking

An internal review can be requested or mandated before certain significant steps
Internal Review: Requested – contract has been flagged for review
Internal Review: Request Withdrawn – request for review has been cancelled
Internal Review: Contract Approved – reviewer approved contract
Internal Review: Self Approved – reviewer approved own contract
Internal Review: Contract Rejected – reviewer rejected contract

Phase 5: Selection

The broker chooses which quote to proceed with
Accepted a Quote – underwriter's quote progressed to firm order
Marked as Firm Order – specific contract instance progressed to firm order
Quote Not Taken Up – quote marked as NTU
Bindable Quote Not Taken Up – bindable line marked as NTU

Phase 6: Firm Order

The broker shows a Firm Order
Changed or Added a Line item – contract heading added and/or text edited in unshown firm order
Deleted contract sections – section heading/s removed from unshown firm order
Requested a Line – firm order made to one or more underwriter/s
Section Assigned to Shown Contract – sections have been added to an unsectioned shown firm order
Showed to Following Market – contract accepted by lead underwriter of a facility/line slip is now being shown to agreement parties of that facility as a firm order

Phase 7: Writing

The underwriter commits to insuring a percentage of the contract
Line Written – specified percentage of contract agreed to
Wrote No Cover Given Line – line written without cover given
Line Percentage Was Revised – specified percentage of contract revised at broker's request
Line Reference Was Revised – references and/or details of agreement revised
Declined Firm Order – contract refused

The underwriter specifies their conditions under which they will insure the contract, and the broker attempts to meet these conditions
Line Conditions/Subjectivities were updated – underwriter has added/revised conditions of acceptance of contract
Subjectivities/Line Conditions Accepted – broker has agreed to conditions
Subjectivities/Line Conditions Rejected – broker has refused conditions
Broker Attempted To Satisfy A Subjectivity – broker has provided a text response to conditions
Added an Attachment from a Subjectivity Response – broker has provided a document response to conditions
Responded to Subjectivity Response – broker has marked conditions as provisionally fulfilled
A Subjectivity Was Removed – underwriter has removed one or more conditions
Stamp Update Submitted – underwriter requests changes to the stamp on a line
Stamp Update Accepted – broker accepts stamp update
Stamp Update Rejected – broker rejects stamp update

The broker needs to alter the agreements confirmed on the contract
Proposed a new Written Line Percentage – broker asks underwriter to modify their percentage of the contract
Line Removal Request – broker requests removal of a line from a contract
Line Removal Accepted – underwriter accepts line removal request
Line Removal Rejected – underwriter rejects line removal request
Off-Platform Line Written – details of deal with an underwriter not in the system recorded
Off-Platform line was adjusted – details of deal with an underwriter not in the system adjusted
Line Removed – Off-Platform Line or (non) notify declaration line removed
Signed Contract reverted to Firm Order – signed contract rolled back to firm order with written lines
Withdrew Firm Order – contract cancelled

The broker needs to alter the contract but the alteration does not warrant an endorsement

Contract Correction Deleted – broker started a contract correction but deleted it unshown.
Contract Correction Shown – broker has shown a contract correction document to one or more underwriters.
Contract Correction Withdrawn – broker has retracted a shown contract correction before underwriter response.
Contract Correction Approved – underwriter has approved a contract correction.
Contract Correction Declined – underwriter has declined a contract correction.
Contract Correction Completed – broker has applied a unanimously-agreed correction to the contract.
Contract Correction Revoked – broker has revoked a contract correction, leaving the contract uncorrected.

Phase 8: Signing

The broker finalises the contract
Signed Lines – contract finalised
Re-Signed Line(s) – contract finalised again after a correction

Phase 9: Endorsing

The broker prepares a contract endorsement to be accepted by underwriters
Cloned Endorsement – created using an old endorsement
Created New Endorsement – created using '+ Endorsement' button
Changed or Added a Line item – contract heading added and/or text edited in draft endorsement
Deleted contract sections – section heading/s removed from draft endorsement
Endorsement was Deleted – unshown draft endorsement deleted

The broker shows the endorsement to underwriters for their acceptance
Showed Endorsement – broker has shown endorsement to underwriter/s
Accepted Endorsement – underwriter has accepted endorsement
Declined Endorsement – underwriter has declined endorsement
Withdrew Endorsement – broker has withdrawn shown endorsement

The broker finalises the endorsement
Bound notify party – non-agreement parties to bound endorsement
Completed Endorsement – endorsement finalised
Mid-Term Participant Change Completed – MTPC endorsement finalised