GUIDEWIRE

Smart Communications Integration with ClaimCenter

A DETAILED DOCUMENT BY GUIDEWIRE PRODUCT DEVELOPMENT

Table of Contents

Overview

This reference implementation provides a fully functional integration between Guidewire applications and Thunderhead. It addresses the following use cases:

- Manual document production
  - DMS storage
  - In-place document content editing
- Document approval process

This reference implementation leverages Guidewire's standard plugin mechanism, Gosu GX Models and Guidewire's messaging framework to interact with the Thunderhead platform. The functional and technical details of how these use cases were implemented are described below.

Who should read this document

Intended audience for this document, as well as the reference implementation:

1. Implementation team integrating the accelerator into the final product.
2. Thunderhead integration managers

What is not included in this reference implementation

Configuration of Thunderhead is not included in this reference implementation. Please refer to Thunderhead's documentation for various configuration topics including template creation, batch configuration, output processing configuration, user access rights, and Thunderhead server and web service deployment.

Integration with Thunderhead's Smart Content Bulk Generator for asynchronous and batch processing is not configured in this version of the reference implementation. However, API support is available for initiating bulk document processing with Thunderhead's Bulk Appliance.

Single Sign-On between ClaimCenter and the Thunderhead Draft Editor is out of scope for this reference implementation.

Applicable Product Versions

This accelerator was implemented and tested with the following products:

1. Thunderhead Smart Communications (in the cloud).
2. ClaimCenter 9.0.0

Terms of Use

We have made this document and accompanying accelerator files available to you as we thought that you might find them useful. As such, they are provided to you "as is", which means that we do not offer you any assurances with respect to them. You will be solely responsible for any changes made to your configuration as a result of you downloading and implementing any of our materials.

You also understand that we own the intellectual property rights to any accelerator documentation/file downloads we make available to you. You must not use them in any way that would adversely affect our rights under applicable law.

Functional Design

Functional Overview

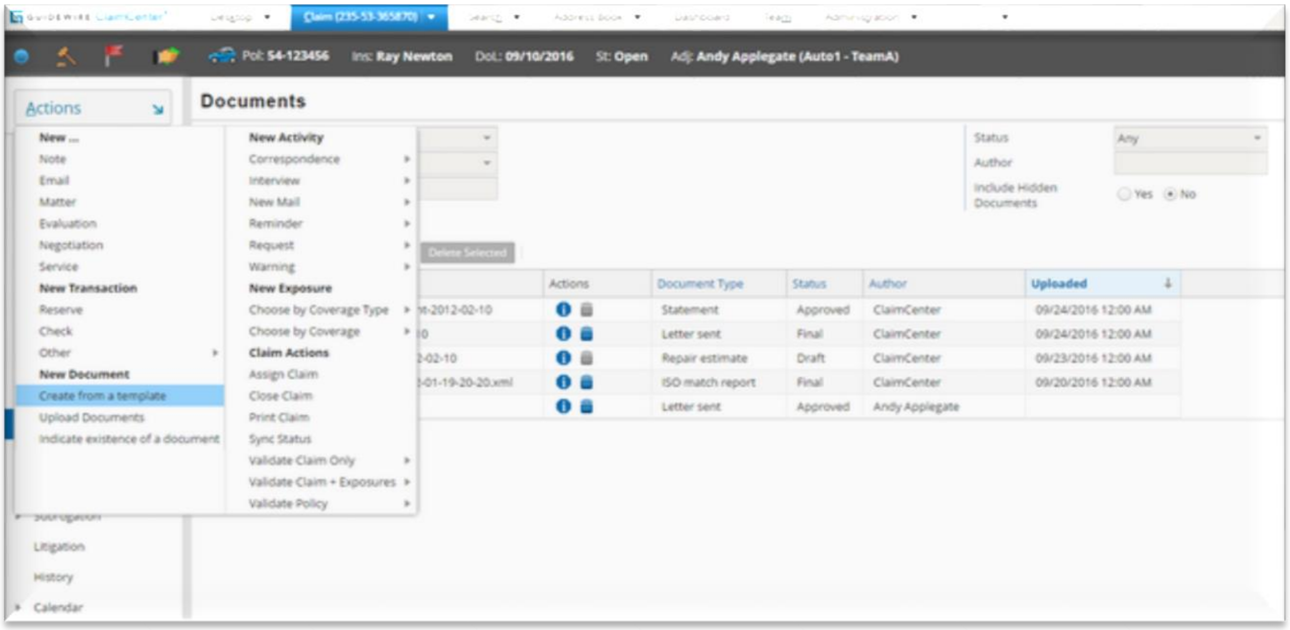
The objective of this reference implementation is to significantly reduce the implementation effort when integrating Thunderhead with ClaimCenter. This section describes the functional objectives of this reference implementation.

Manual Document Production

In certain scenarios, ClaimCenter users may need to initiate manual creation of a document. In such cases, the user would navigate to the New Document screen and choose a template. The steps below (with screenshots) demonstrate this case. The sample Ray Newton claim is used as a starting point.

Completing the below scenario as superuser will bypass the need for document approval, as superuser has sufficient permissions. For an outline of the document approval process, see the next section Document Approval Process.

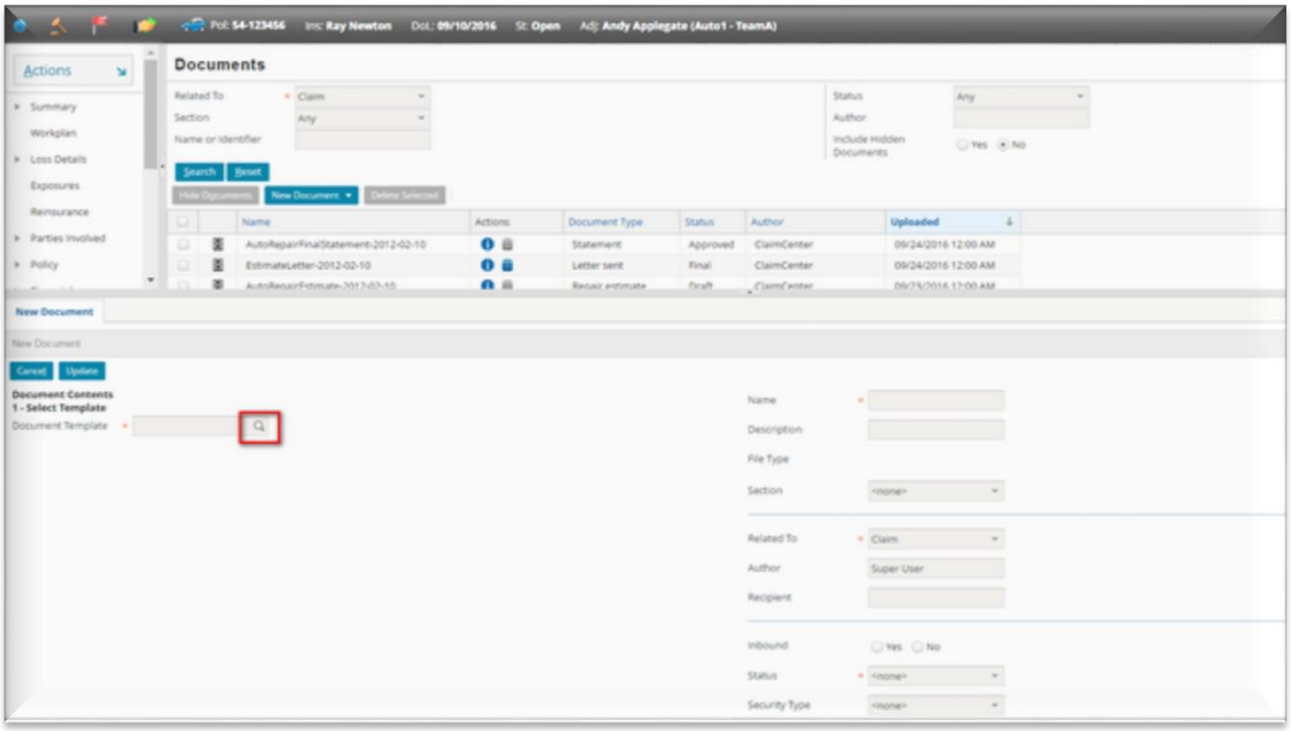
1. From the "Actions" menu, select the option "Create from a template":



Status: Any
Author:
Include Hidden Documents: Yes [✓ No]

	Actions	Document Type	Status	Author	Uploaded
nt-2012-02-10		Statement	Approved	ClaimCenter	09/24/2016 12:00 AM
10		Letter sent	Final	ClaimCenter	09/24/2016 12:00 AM
2-02-10		Repair estimate	Draft	ClaimCenter	09/23/2016 12:00 AM
-01-19-20-20.xml		ISO match report	Final	ClaimCenter	09/20/2016 12:00 AM
		Letter sent	Approved	Andy Applegate	

2. Now press the magnifier icon to select the document template:



Documents

Related To *	Claim
Section	Any
Name or Identifier	
Status	Any
Author	
Include Hidden Documents	Yes [✓ No]

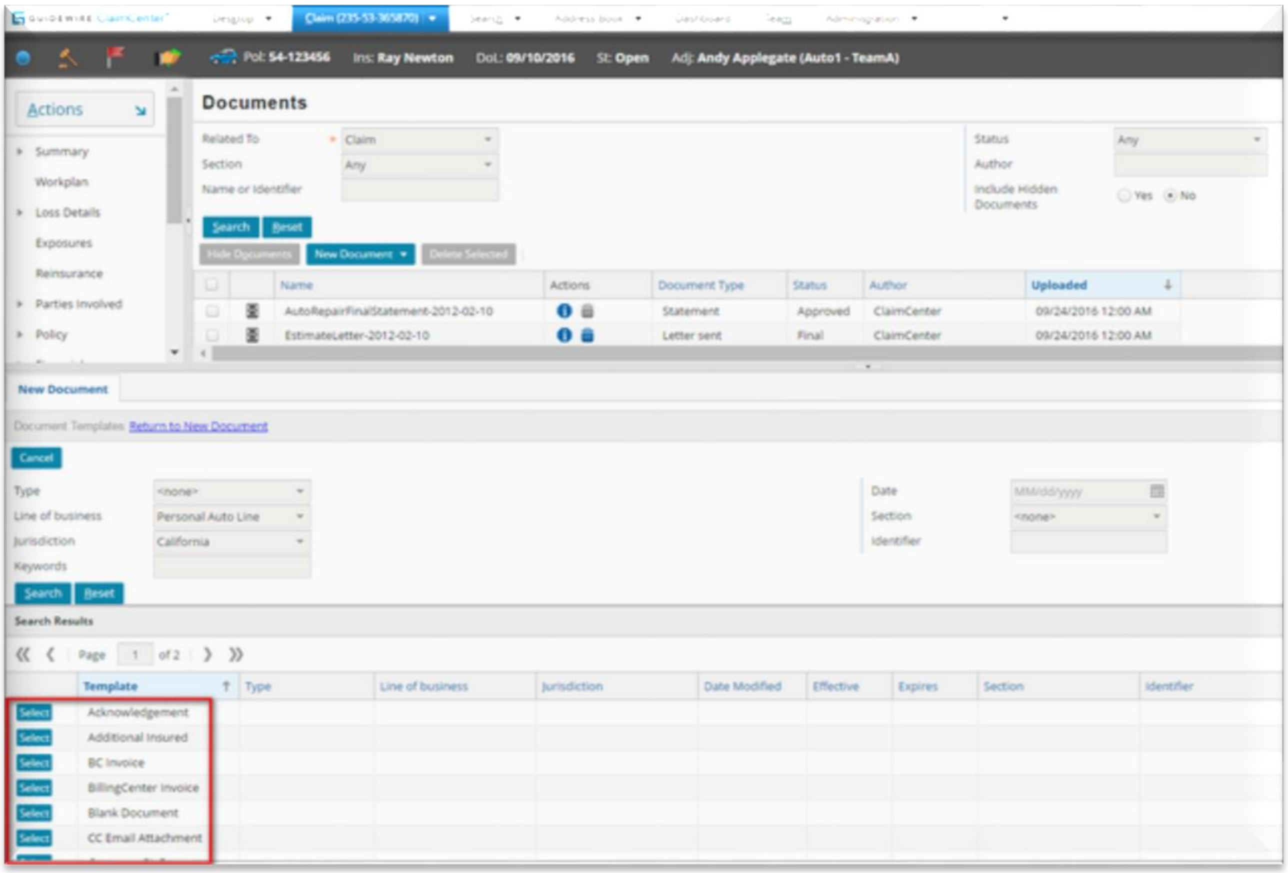
Name	Actions	Document Type	Status	Author	Uploaded
AutoRepairFinalStatement-2012-02-10		Statement	Approved	ClaimCenter	09/24/2016 12:00 AM
EstimateLetter-2012-02-10		Letter sent	Final	ClaimCenter	09/24/2016 12:00 AM
AutoRepairEstimate-2012-02-10		Repair estimate	Draft	ClaimCenter	09/23/2016 12:00 AM

New Document

Document Contents
1 - Select Template

Document Template *	
Name *	
Description	
File Type	
Section	<none>
Related To *	Claim
Author	Super User
Recipient	
Inbound	Yes No
Status *	<none>
Security Type	<none>

3. This popup shows all the templates configured in the Thunderhead platform; select one:



Pol: 54-123456 Ins: Ray Newton DoL: 09/10/2016 St: Open Adj: Andy Applegate (Auto1 - TeamA)

Documents

Related To: Claim
Section: Any
Name or Identifier:
Status: Any
Author:
Include Hidden Documents: Yes [✓ No]

Name	Actions	Document Type	Status	Author	Uploaded
AutoRepairFinalStatement-2012-02-10		Statement	Approved	ClaimCenter	09/24/2016 12:00 AM
EstimateLetter-2012-02-10		Letter sent	Final	ClaimCenter	09/24/2016 12:00 AM

Document Templates Return to New Document

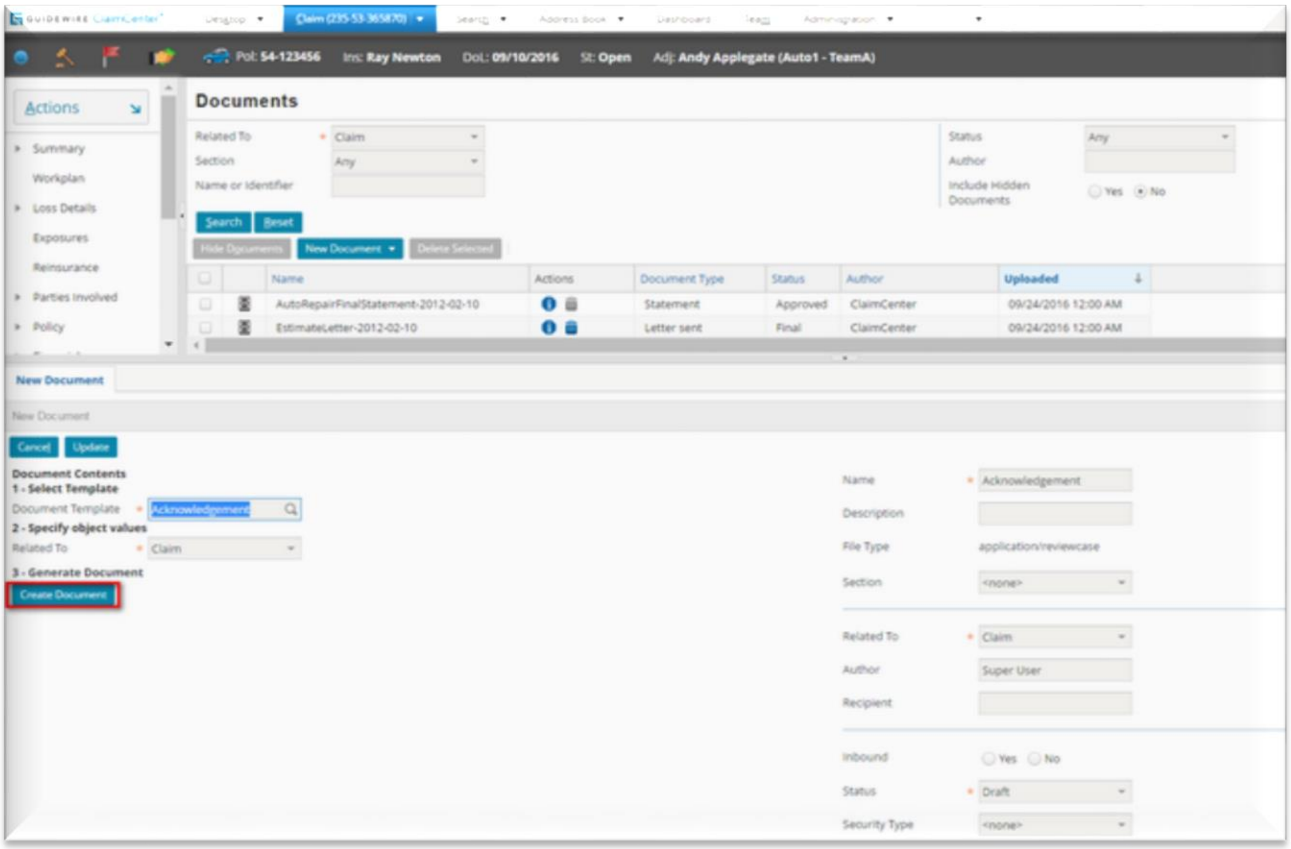
Type: <none>
Line of business: Personal Auto Line
Jurisdiction: California
Keywords:
Date: MM/dd/yyyy
Section: <none>
Identifier:

Search Results

Page 1 of 2

	Template	Type	Line of business	Jurisdiction	Date Modified	Effective	Expires	Section	Identifier
Select	Acknowledgement								
Select	Additional Insured								
Select	BC Invoice								
Select	BillingCenter Invoice								
Select	Blank Document								
Select	CC Email Attachment								

4. After selecting the template, a document can be created using the “Create Document” button. Note the document status is Draft.



Pol: 54-123456 Ins: Ray Newton DoL: 09/10/2016 St: Open Adj: Andy Applegate (Auto1 - TeamA)

Documents

Related To	* Claim
Section	Any
Name or Identifier	
Status	Any
Author	
Include Hidden Documents	Yes [✓ No]

	Name	Actions	Document Type	Status	Author	Uploaded
	AutoRepairFinalStatement-2012-02-10		Statement	Approved	ClaimCenter	09/24/2016 12:00 AM
	EstimateLetter-2012-02-10		Letter sent	Final	ClaimCenter	09/24/2016 12:00 AM

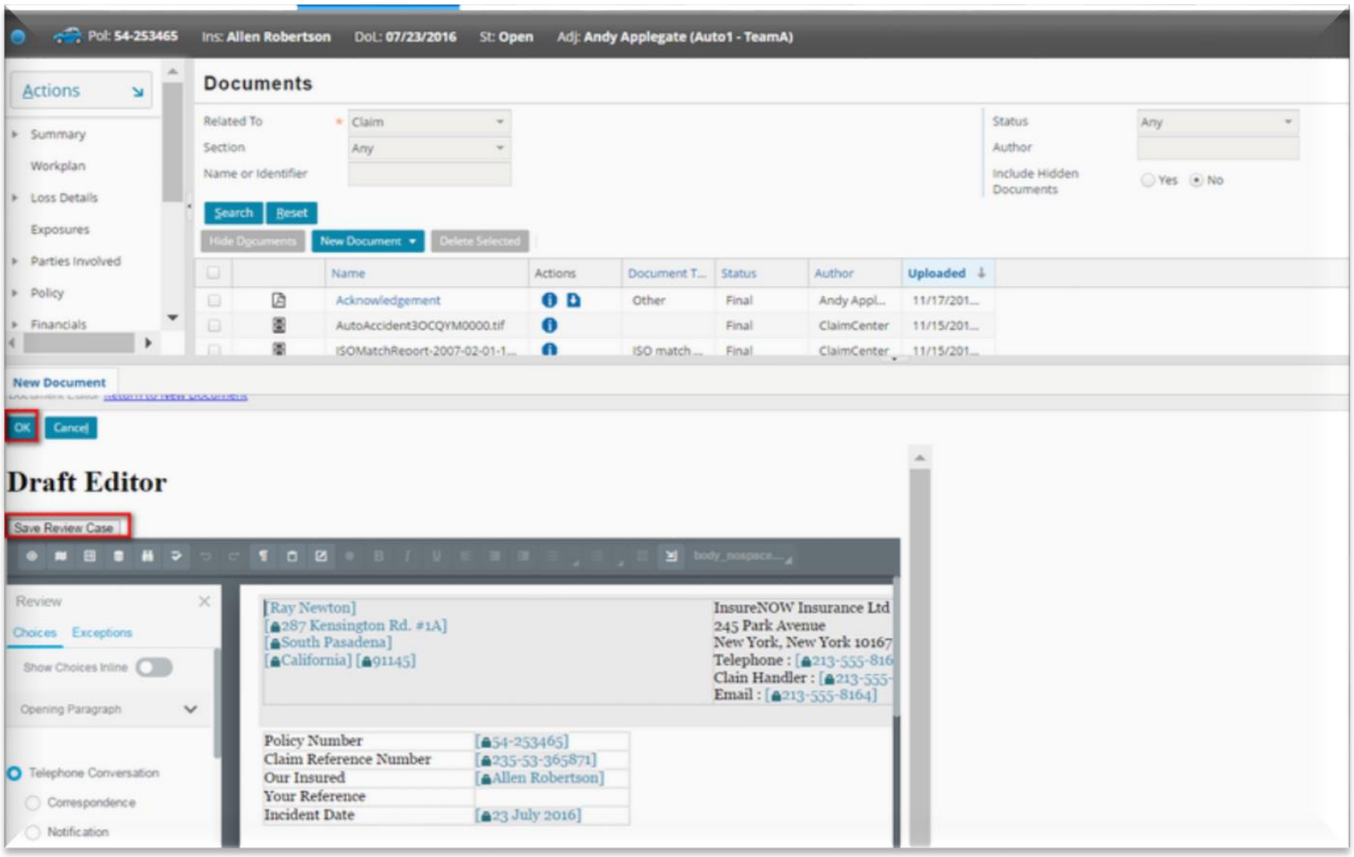
New Document

New Document

Document Contents
1 - Select Template
Document Template * Acknowledgement
2 - Specify object values
Related To * Claim
3 - Generate Document

Name	* Acknowledgement
Description	
File Type	application/reviewcase
Section	<none>
Related To	* Claim
Author	Super User
Recipient	
Inbound	Yes No
Status	* Draft
Security Type	<none>

5. When using Thunderhead Smart Communications you will see a JavaScript control called the Draft Editor, which allows you to edit the document. Note you will initially have to sign into the Draft editor. When finished editing press the **Save Review Case** and then the **OK**. Button.



Pol: 54-253465 Ins: Allen Robertson DoL: 07/23/2016 St: Open Adj: Andy Applegate (Auto1 - TeamA)

Documents

Related To: Claim
Section: Any
Name or Identifier:
Status: Any
Author:
Include Hidden Documents: Yes [✓ No]

		Name	Actions	Document T...	Status	Author	Uploaded
		Acknowledgement		Other	Final	Andy Appl...	11/17/201...
		AutoAccident3OCQYM0000.tif			Final	ClaimCenter	11/15/201...
		ISOMatchReport-2007-02-01-1...		ISO match ...	Final	ClaimCenter	11/15/201...

Draft Editor

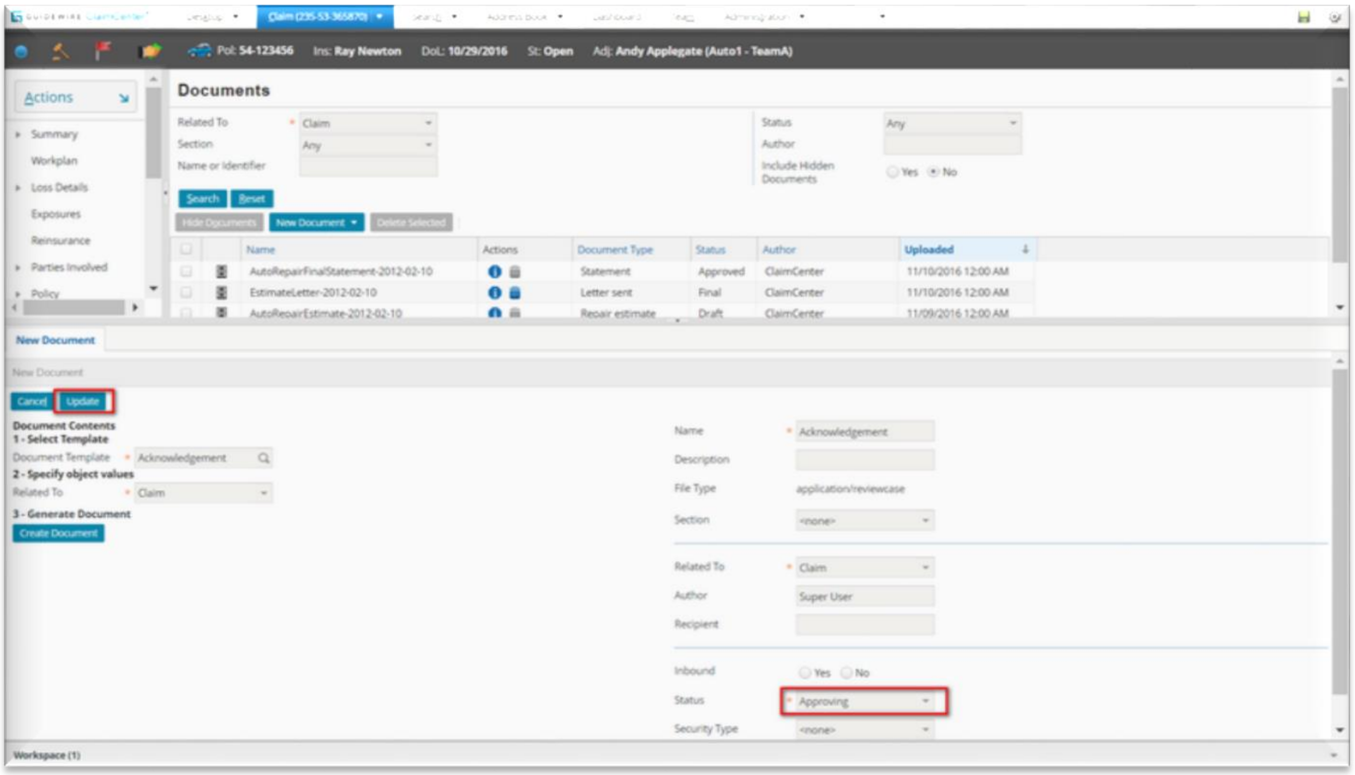
Review — Choices, Exceptions
Show Choices Inline
Opening Paragraph
- [✓ Telephone Conversation]
- Correspondence
- Notification

[Ray Newton]
[287 Kensington Rd. #1A]
[South Pasadena]
[California] [91145]

InsureNOW Insurance Ltd
245 Park Avenue
New York, New York 10167
Telephone : [213-555-816
Clain Handler : [213-555-
Email : [213-555-8164]

Policy Number	[54-253465]
Claim Reference Number	[235-53-365871]
Our Insured	[Allen Robertson]
Your Reference	
Incident Date	[23 July 2016]

6. When the document editor closes, set status to “Approving” and click “Update”



Pol: 54-123456 Ins: Ray Newton DoL: 10/29/2016 St: Open Adj: Andy Applegate (Auto1 - TeamA)

Documents

Related To: Claim
Section: Any
Name or Identifier:
Status: Any
Author:
Include Hidden Documents: Yes [✓ No]

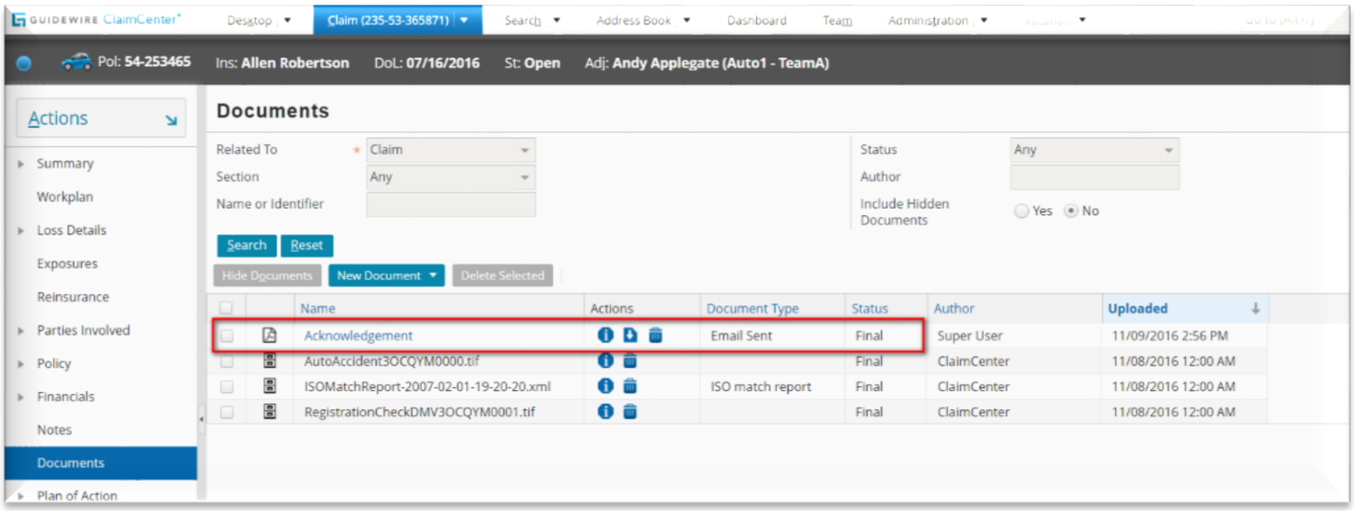
	Name	Actions	Document Type	Status	Author	Uploaded
	AutoRepairFinalStatement-2012-02-10		Statement	Approved	ClaimCenter	11/10/2016 12:00 AM
	EstimateLetter-2012-02-10		Letter sent	Final	ClaimCenter	11/10/2016 12:00 AM
	AutoRepairEstimate-2012-02-10		Repair estimate	Draft	ClaimCenter	11/09/2016 12:00 AM

New Document

Document Contents
1 - Select Template
Document Template: Acknowledgement
2 - Specify object values
Related To: Claim
3 - Generate Document

Name: Acknowledgement
Description:
File Type: application/reviewcase
Section: <none>
Related To: Claim
Author: Super User
Recipient:
Inbound: Yes No
Status: Approving
Security Type: <none>

This creates a new document with a status of “Final” as you are logged in as superuser no approval is required. For more details on the approval process, see the next section Document Approval Process.



Pol: 54-253465 Ins: Allen Robertson DoL: 07/16/2016 St: Open Adj: Andy Applegate (Auto1 - TeamA)

Documents

Related To: Claim
Section: Any
Name or Identifier:
Status: Any
Author:
Include Hidden Documents: Yes [✓ No]

	Name	Actions	Document Type	Status	Author	Uploaded
	Acknowledgement		Email Sent	Final	Super User	11/09/2016 2:56 PM
	AutoAccident3OCQYM0000.tif			Final	ClaimCenter	11/08/2016 12:00 AM
	ISOMatchReport-2007-02-01-19-20-20.xml		ISO match report	Final	ClaimCenter	11/08/2016 12:00 AM
	RegistrationCheckDMV3OCQYM0001.tif			Final	ClaimCenter	11/08/2016 12:00 AM

Document Approval Process

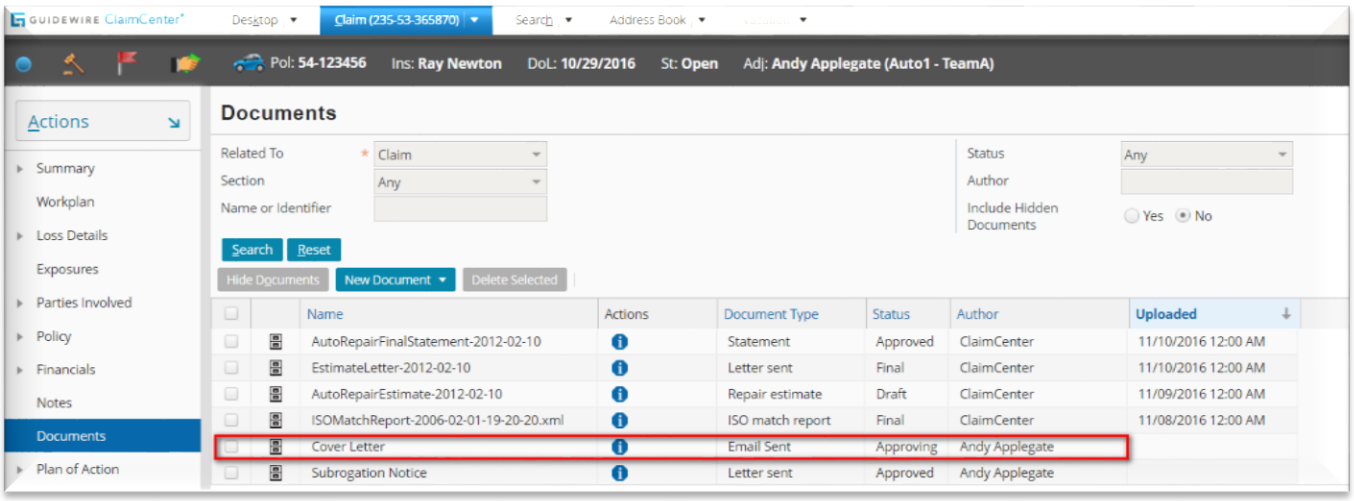
For manual document creation, the draft document may go through several edits before it is ready for production and distribution. Once the document is ready to be distributed, the user can submit the document for review by a supervisor. At this time, the document status changes from “Draft” to “Approving”.

The application will then create an approval activity for a supervisor. Once the Supervisor approves the document, its status is set to “Approved”. This triggers a document change event, which in turn sends the document to Thunderhead via the Thunderhead message destination if the document’s SendOnApproval_Ext flag is set to true.

A new system permission exists “docapproval” in the system permission typelist. Lack of this permission prevents users approving their own documents. Adding it to a user allows them to approve their own documents.

The following scenario assumes that Andy Applegate cannot approve his own documents.

1. First, login as Andy Applegate and create a document as outlined in the section above, *Manual Document Production*. At the final stage of document creation, step 6 above, submit the document for approval. This results in the document having a status of “Approving”.



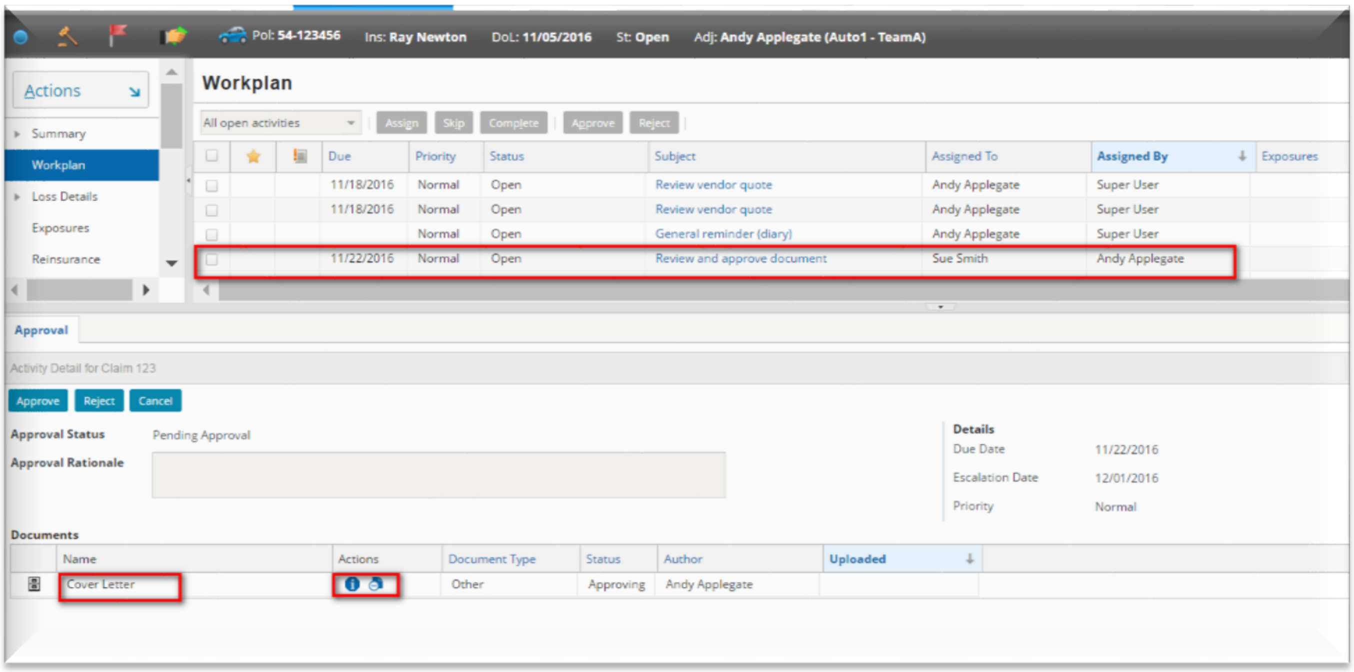
Pol: 54-123456 Ins: Ray Newton DoL: 10/29/2016 St: Open Adj: Andy Applegate (Auto1 - TeamA)

Documents

Related To: Claim
Section: Any
Name or Identifier:
Status: Any
Author:
Include Hidden Documents: Yes [✓ No]

Name	Actions	Document Type	Status	Author	Uploaded
AutoRepairFinalStatement-2012-02-10		Statement	Approved	ClaimCenter	11/10/2016 12:00 AM
EstimateLetter-2012-02-10		Letter sent	Final	ClaimCenter	11/10/2016 12:00 AM
AutoRepairEstimate-2012-02-10		Repair estimate	Draft	ClaimCenter	11/09/2016 12:00 AM
ISOMatchReport-2006-02-01-19-20-20.xml		ISO match report	Final	ClaimCenter	11/08/2016 12:00 AM
Cover Letter		Email Sent	Approving	Andy Applegate	
Subrogation Notice		Letter sent	Approved	Andy Applegate	

2. Login as Sue Smith and open the Claims Workplan screen. It shows the newly created document review activity, assigned to the supervisor Sue Smith. Opening the activity shows: the linked document, provides actions for Approving or Rejecting the activity, and an area for entering an approval or rejection rational.



Pol: 54-123456 Ins: Ray Newton DoL: 11/05/2016 St: Open Adj: Andy Applegate (Auto1 - TeamA)

Workplan

Due	Priority	Status	Subject	Assigned To	Assigned By	Exposures
11/18/2016	Normal	Open	Review vendor quote	Andy Applegate	Super User	
11/18/2016	Normal	Open	Review vendor quote	Andy Applegate	Super User	
	Normal	Open	General reminder (diary)	Andy Applegate	Super User	
11/22/2016	Normal	Open	Review and approve document	Sue Smith	Andy Applegate	

Activity Detail for Claim 123

Approval Status: Pending Approval

Approval Rationale:

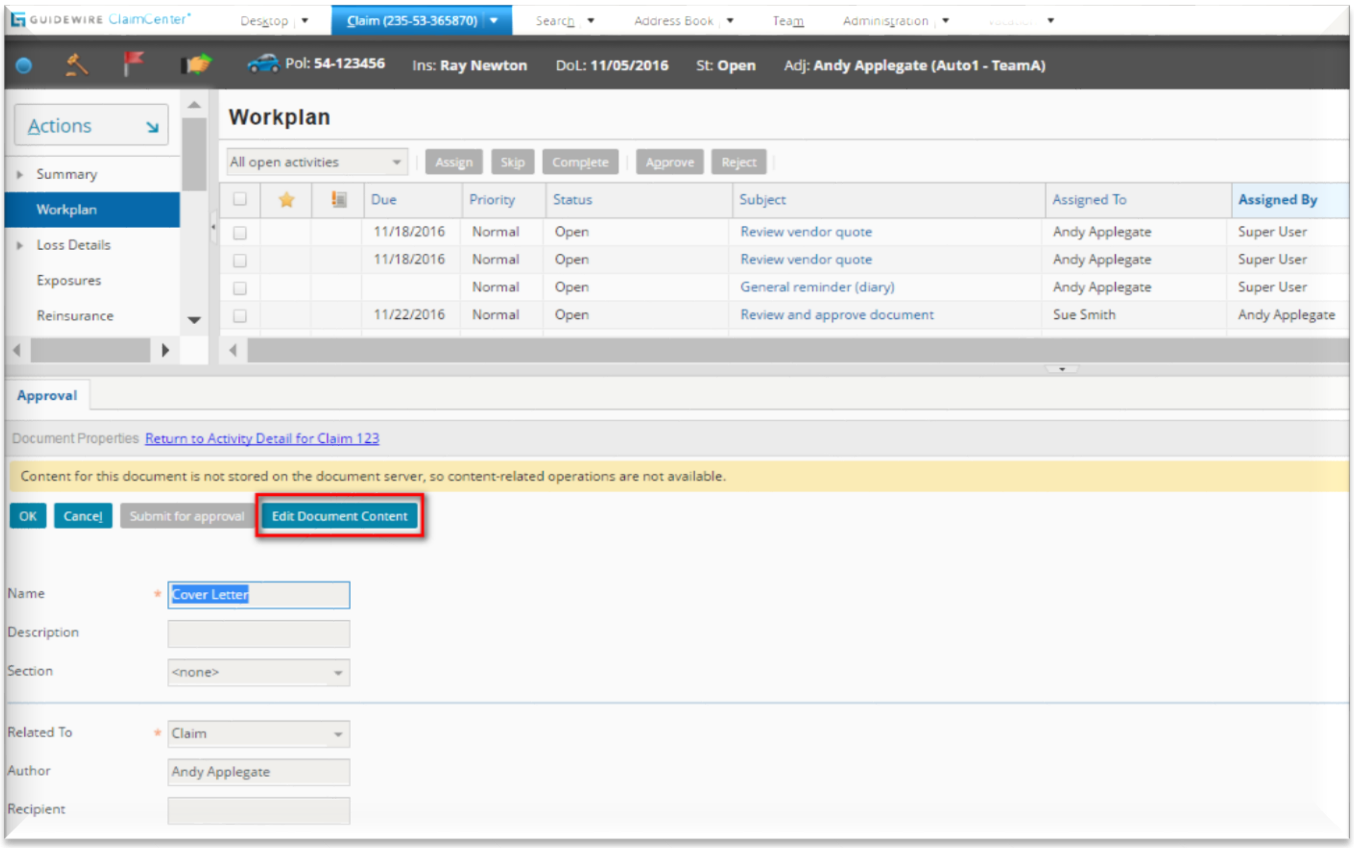
Details

Due Date	11/22/2016
Escalation Date	12/01/2016
Priority	Normal

Documents

Name	Actions	Document Type	Status	Author	Uploaded
Cover Letter		Other	Approving	Andy Applegate	

Note clicking on the "**i**" icon in the **Actions** column above and then clicking on the **Edit** button which appears will bring you to the following screen . From here you can view or edit the document before approving.



Pol: 54-123456 Ins: Ray Newton DoL: 11/05/2016 St: Open Adj: Andy Applegate (Auto1 - TeamA)

Workplan

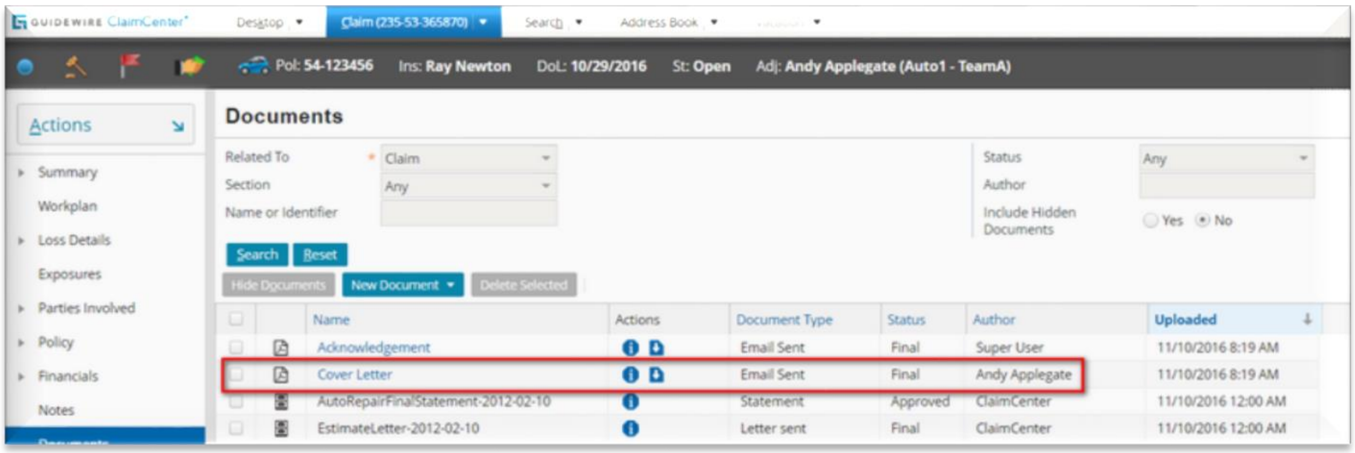
Due	Priority	Status	Subject	Assigned To	Assigned By
11/18/2016	Normal	Open	Review vendor quote	Andy Applegate	Super User
11/18/2016	Normal	Open	Review vendor quote	Andy Applegate	Super User
	Normal	Open	General reminder (diary)	Andy Applegate	Super User
11/22/2016	Normal	Open	Review and approve document	Sue Smith	Andy Applegate

Document Properties Return to Activity Detail for Claim 123

Content for this document is not stored on the document server, so content-related operations are not available.

Name *	Cover Letter
Description	
Section	<none>
Related To *	Claim
Author	Andy Applegate
Recipient	

3. **Approving** the document will result in the status changing to “Approved”. Event fired rules will then pick up the document and send it to Thunderhead for production. Once this is complete, the status changes to “Final” and the generated PDF is stored in the Document Management System.



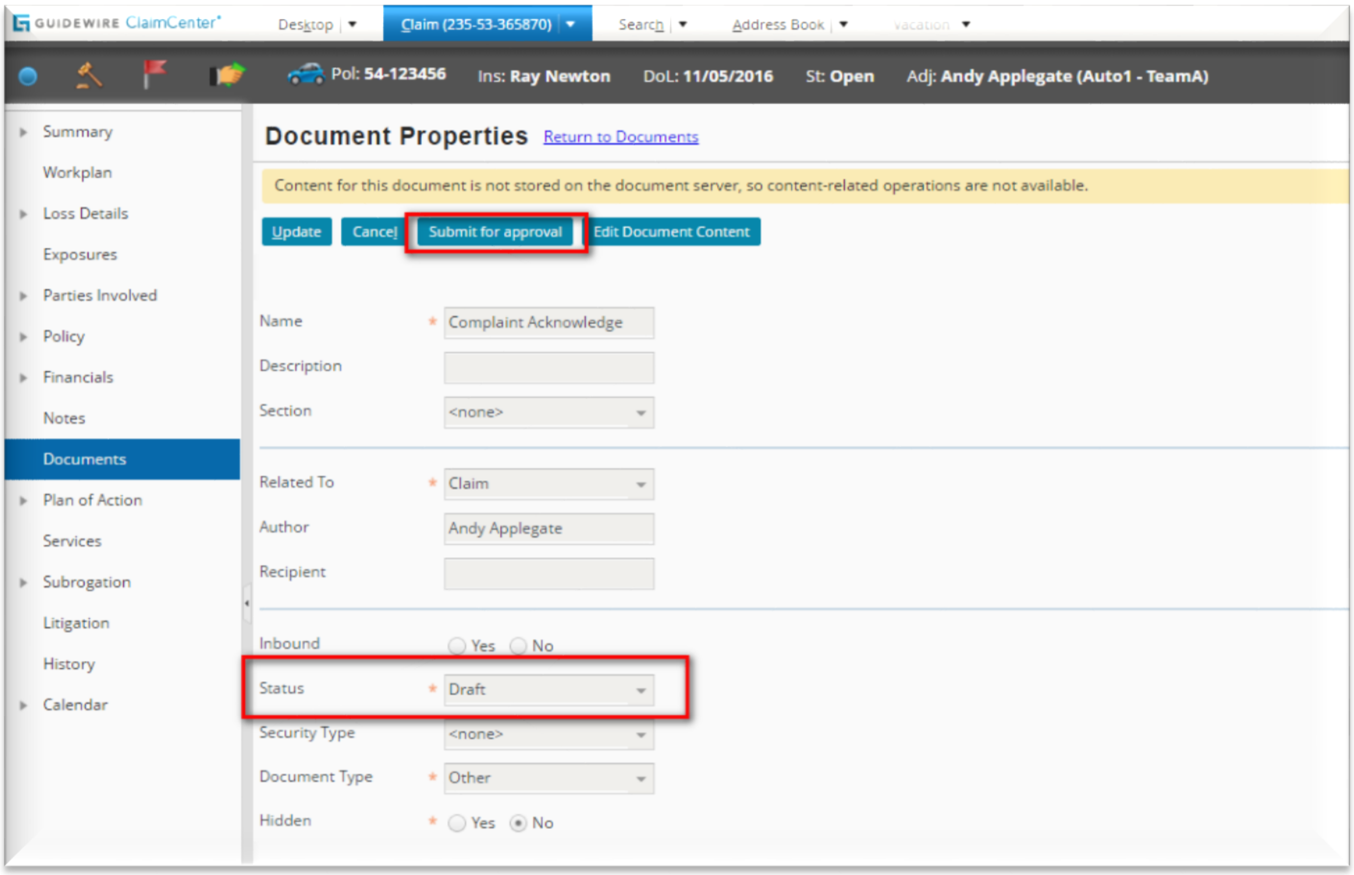
Pol: 54-123456 Ins: Ray Newton DoL: 10/29/2016 St: Open Adj: Andy Applegate (Auto1 - TeamA)

Documents

Related To: Claim
Section: Any
Name or Identifier:
Status: Any
Author:
Include Hidden Documents: Yes [✓ No]

Name	Actions	Document Type	Status	Author	Uploaded
Acknowledgement		Email Sent	Final	Super User	11/10/2016 8:19 AM
Cover Letter		Email Sent	Final	Andy Applegate	11/10/2016 8:19 AM
AutoRepairFinalStatement-2012-02-10		Statement	Approved	ClaimCenter	11/10/2016 12:00 AM
EstimateLetter-2012-02-10		Letter sent	Final	ClaimCenter	11/10/2016 12:00 AM

4. **Rejecting** the document results in the status being change back to draft. Resubmitting the document for approval can be done from the Document Properties screen:



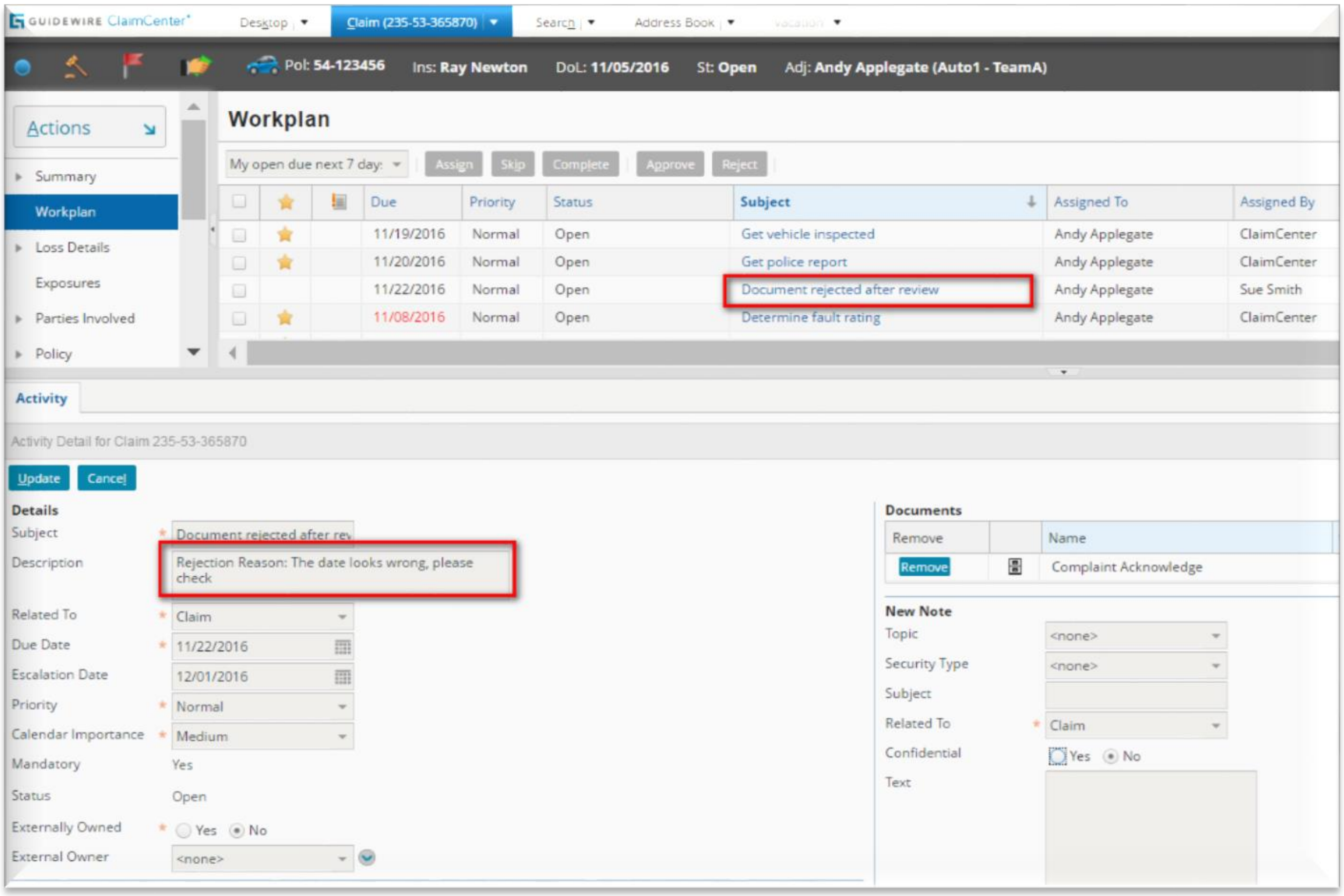
Pol: 54-123456 Ins: Ray Newton DoL: 11/05/2016 St: Open Adj: Andy Applegate (Auto1 - TeamA)

Document Properties Return to Documents

Content for this document is not stored on the document server, so content-related operations are not available.

Name *	Complaint Acknowledge
Description	
Section	<none>
Related To *	Claim
Author	Andy Applegate
Recipient	
Inbound	Yes No
Status *	Draft
Security Type	<none>
Document Type *	Other
Hidden *	Yes [✓ No]

5. **Rejecting** the document also creates a review activity. The activities description field shows the reason for the documents rejection, copied from the supervisors' approval activity. The activity serves as a reminder that the document needs to be reviewed and resubmitted.



Pol: **54-123456** Ins: **Ray Newton** DoL: **11/05/2016** St: **Open** Adj: **Andy Applegate (Auto1 - TeamA)**

Workplan

My open due next 7 day:

Due	Priority	Status	Subject	Assigned To	Assigned By
11/19/2016	Normal	Open	Get vehicle inspected	Andy Applegate	ClaimCenter
11/20/2016	Normal	Open	Get police report	Andy Applegate	ClaimCenter
11/22/2016	Normal	Open	Document rejected after review	Andy Applegate	Sue Smith
11/08/2016	Normal	Open	Determine fault rating	Andy Applegate	ClaimCenter

Activity Detail for Claim 235-53-365870

Details

Subject	Document rejected after rev
Description	Rejection Reason: The date looks wrong, please check
Related To	Claim
Due Date	11/22/2016
Escalation Date	12/01/2016
Priority	Normal
Calendar Importance	Medium
Mandatory	Yes
Status	Open
Externally Owned	Yes [✓ No]
External Owner	<none>

Documents

Remove		Name
Remove		Complaint Acknowledge

New Note

Topic	<none>
Security Type	<none>
Subject	
Related To	Claim
Confidential	Yes [✓ No]
Text	

Technical Design

This technical overview of automated document creation using Thunderhead assumes the reader has knowledge of the following:

Messaging

Rules

Web Services

Activities and Approval Activities

Integration Overview

This reference implementation was designed in such a way that the integration transport can be changed via configuration.

Message Destination

The reference implementation defines a new messaging destination called Thunderhead. It is configured in messaging-config.xml. The destination is invoked when a document is added or changed. The message listener, ThunderheadTransport, determines how ClaimCenter invokes Thunderhead.

Web Services and the IThunderheadService Interface

Web services communication with Thunderhead is performed through implementations of two interfaces: CMSAPIWebWrapped and JobAPIWebWrapped. The Spring framework injects the appropriate implementation, determined by a configuration parameter described below, into the IThunderheadService implementation class.

JMS

The other integration transport (JMS) is initiated by the DocumentJmsTransport, which can be configured to listen for messages on the Thunderhead message transport.

Manual document creation

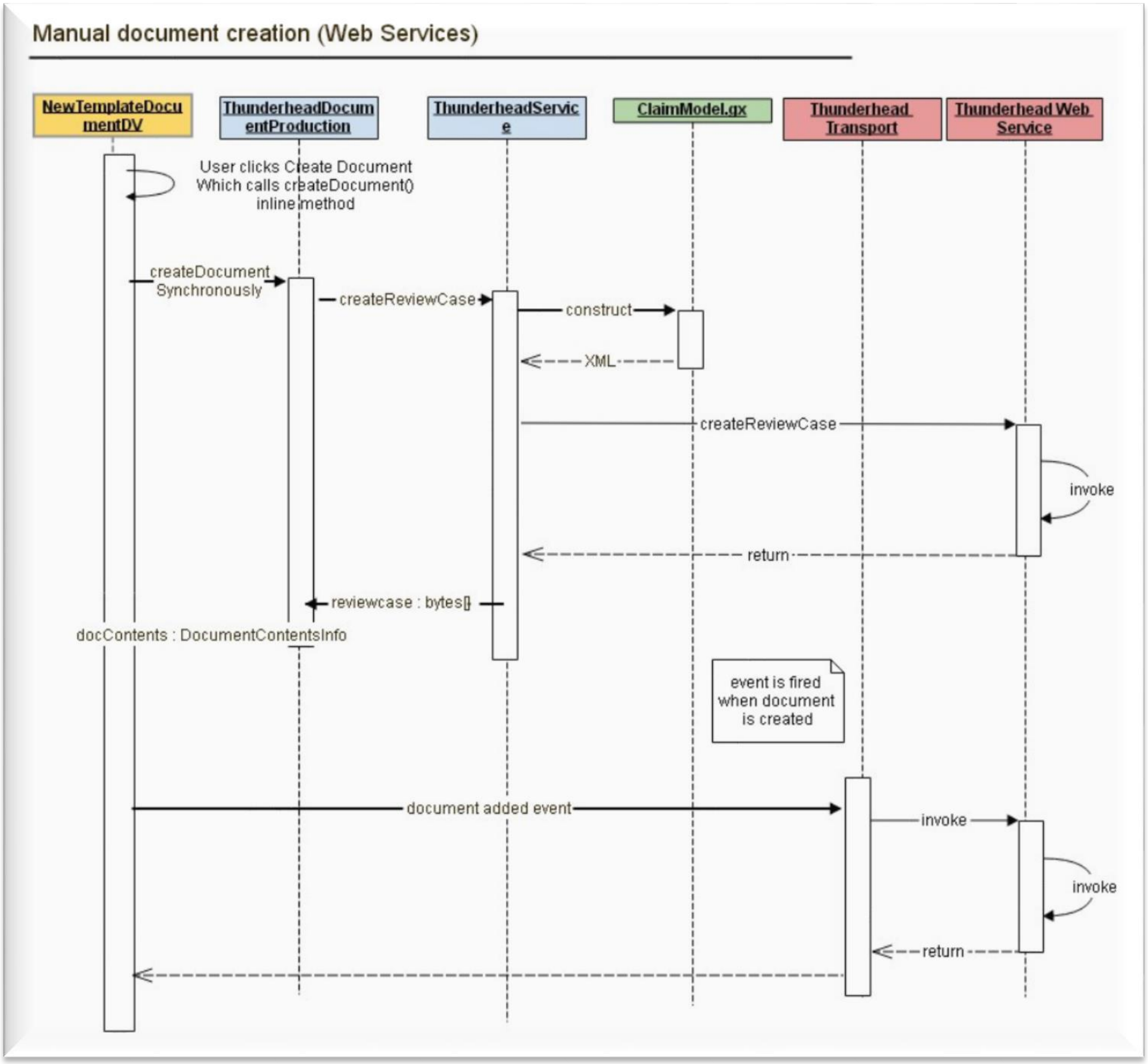

Manual document creation (Web Services)
NewTemplateDocumentDV
ThunderheadDocumentProduction
ThunderheadService
ClaimModel.gx
Thunderhead Transport
Thunderhead Web Service
User clicks Create Document Which calls createDocument() inline method
createDocument Synchronously
createReviewCase
construct
XML
createReviewCase
invoke
return
reviewcase : bytes[]
docContents : DocumentContentsInfo
event is fired when document is created
document added event
invoke
invoke
return

Figure 1: Manual document creation

The manual document creation flow begins when the user clicks on the createDocument button on the NewTemplateDocumentDV. This button click calls a method called createDocument(). This method does a number of things, but the main action is to call the ThunderheadDocumentProduction.createDocumentSynchronously method.

ThunderheadDocumentProduction.createDocumentSynchronously calls ThunderheadService.createReviewCase, which makes a web service call to Thunderhead to create a review case, which is what Thunderhead will eventually use to issue documents. A ReviewCase_Ext is created and linked to the Document entity. The review case is then returned to be passed to the Thunderhead document editor for display to the user.

When the document status is switched to approved, the event fired rule will receive the document changed or added event and start the messaging procedure to send the document to Thunderhead for issuance.

XML Payload Production

ClaimCenter dynamically generates transaction XML based on entities in the ClaimCenter database, and Thunderhead merges the transaction data with a predefined template to generate the document content. Because of the cost associated with generating transaction data and the size of the resulting payload, it is a best practice when working with Thunderhead to limit the amount of data included in the payload whenever possible. A contact letter or a request for information from a policyholder may not require the entire claim "graph," for example. Part of the development activities around Thunderhead integration will be to define the set of models required for different document production use cases.

This reference implementation uses GX Models to produce XML from a Claim. The claim is passed to the ClaimModel.gx GX Model.

Spring Integration

This accelerator makes use of the Spring Gosu Integration Reference Implementation[1]. This accelerator allows the implementer to use some of the features of the Spring Framework[2]. In this reference implementation, we make use of the Dependency Injection (DI) functionality of Spring. Spring DI creates the web service wrapper classes and injects them into the ThunderheadService class. Spring DI is also used in testing to mock the connection to the Thunderhead web services.

Integration Components

Thunderhead Smart Communications Web Services

Thunderhead Smart Communications provides RESTful web services for integration. All of these RESTful services are synchronous. Thunderhead does provide a separate application that can be used to provide batch processing of documents but this version of the reference implementation does not integrate with this application. This reference implementation makes use of the following RESTful services provided by Thunderhead Smart Communications:

- CMS API
- Correspondence API
- Job API

For more information about these APIs please visit the Thunderhead Smart Communications documentation.

[1] https://guidewire.hivelive.com/posts/1ebfd1d652

[2] http://spring.io

Messaging Infrastructure

This reference implementation is able to send batch jobs to Thunderhead in two ways: web services, and using Thunderhead's JMS queues. Two plugins are added to ClaimCenter to implement this: ThunderheadTransport and ThunderheadReply.

SOAP implementation:

DocumentWebTransport (implements MessageTransport interface)

This plugin (together with events being fired) makes sure that the batch jobs are sent to the Thunderhead system using its Web Services API. The batches are then processed asynchronously by Thunderhead.

DocumentWebReply (implements MessageReply interface)

Because of the asynchronous character of batch processing in the Thunderhead platform, this reply plugin checks the status of the batch jobs sent previously by the **DocumentWebTransport** plugin. Internally, it uses the MessageProcessor class, which hides the details of creating the worker thread responsible for executing periodic actions polling for updates.

JMS implementation:

DocumentJMSTransport (implements MessageTransport interface)

This plugin (together with events being fired) makes sure that the batch jobs are sent to the Thunderhead system using its JMS input queue. The batches are then processed asynchronously by Thunderhead.

DocumentJMSReply (implements MessageReply interface)

This plugin handles the notifications from Thunderhead about the batch job status changes. Internally, it is implemented as a listener for Thunderhead's JMS output queue.

RESTful implementation:

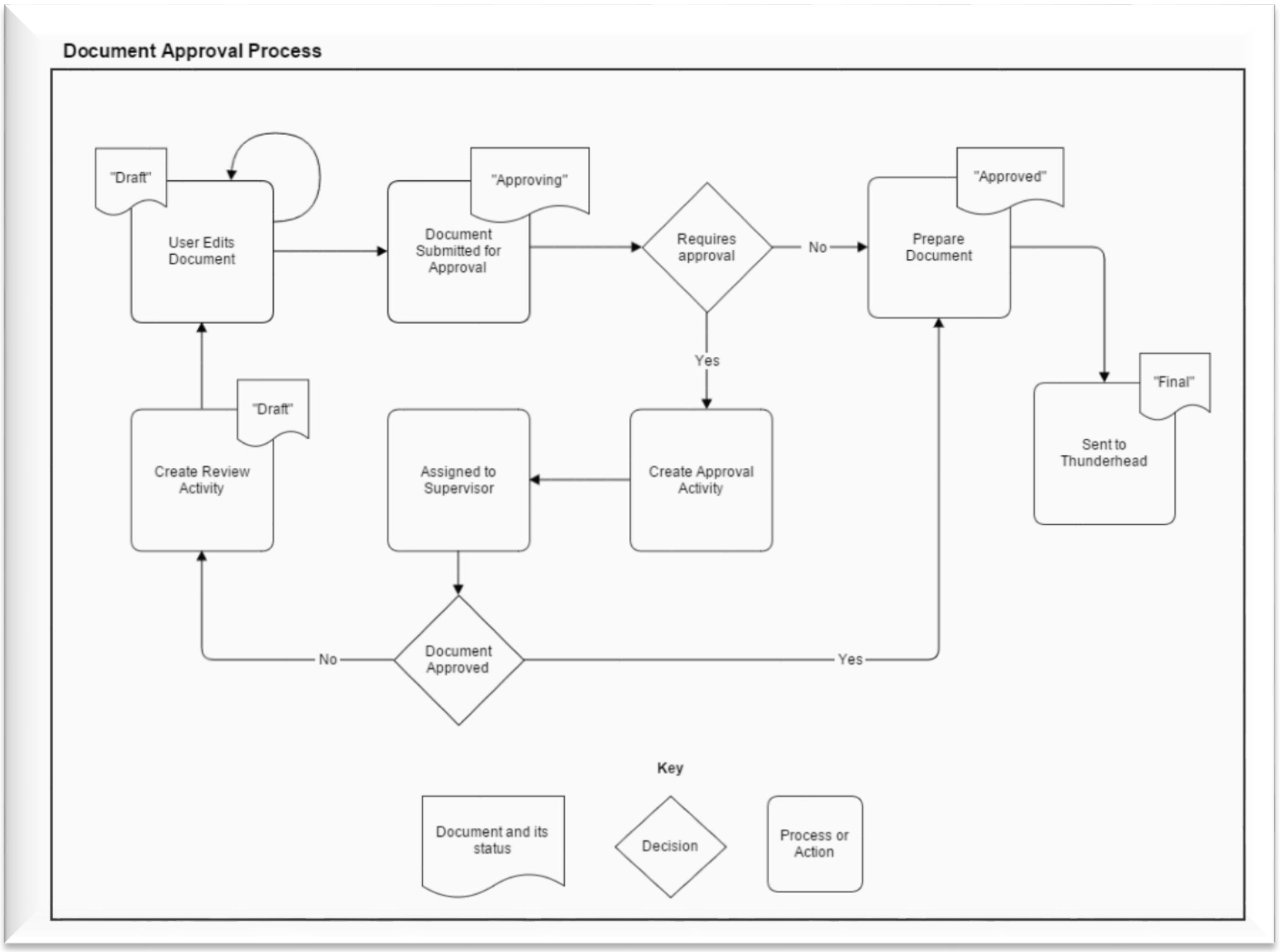
DocumentRestTransport (implements MessageTransport interface)

This plugin (together with events being fired) makes sure that the document requests are being sent to the Thunderhead Smart Communications system using its RESTful API. There is no reply plugin because Thunderhead Smart Communications is synchronous so there is no Reply Transport needed.

Document Approval Process

The below diagram *Document Approval Process* outlines the high-level steps involved in approving a document. The follow sections outline specific implementation details.

- Document Approval Required
- Approval Activity Assignment
- Document Approved Decision
- Review Activity Assignment
- Prepare Document for Production



Document Approval Required

Submitting a document for approval, i.e. changing its status from "Draft" to "Approving", triggers the document preupdate rule, *"DPU01000 - Document Approval"* rule.

The rule has two child rules, "*DPU01100 - User Has Approval Permission*" and *"DPU01200 - User Does Not Have Approval Permission"*. The first of which, verifies the user has the required permission *"docapproval"* and sets the document status to approved.

The second rule verifies that the user lacks the "*docapproval*" permission and creates the approval activity, leaving the document status as "Approving"

Approval Activity Assignment

The Default Group Activity Assignment rule *"DGA07500 - Document Approval Assignment"* and the Global Activity Assignment Rule *"GAA06500 - Document Approval Activity"* are responsible for assigning the approval activity.

By default, the activity is assigned to the same group as the claims assigned group and specifically to the supervisor of that group.

Document Approved Decision

The OOTB functionality for approval activities handles the activity approval/rejection logic. A new worksheet, *"DocumentApprovalDetailWorksheet.pcf"* was created to allow the reviewer to enter feedback on the document.

Two activity preupdate rules, *"APU01000 - Document Approval Activity Approved"* and *"APU02000 - Document Approval Activity Rejected"* handle the approval/rejection of the activity respectively.

Approving the activity sets the documents status to "Approved".

Rejecting the activity creates a review activity and the documents status is set back to "Draft".

Review Activity Assignment

The Default Group Activity Assignment rule *"DGA07600 - Document Rejection Activity"* and the Global Activity Assignment Rule *"GAA06500 - Document Approval Activity"* are responsible for assigning the review activity.

By default, the activity is assigned to the creator of the document.

Prepare Document for Production

Approved documents are picked up by the Event Fired Rule, *"Thunderhead – Submit"* and processed as described in the previous sections.

Data Model Extensions

Data Model extensions are described in the Deployment section in this document.

User Interface Modifications

User Interface modifications are described in the Deployment section in this document.

Other Resources

These resources and their exact location is also listed in the Deployment section in this document.

Deployment

This section contains a step-by-step process for deploying and running the reference implementation. The reference implementation is distributed as a zip that contains the following folders:

1. **.idea** – contains updated SDK definition file with JAR references required to compile and run the xCenter product from Studio
2. **configuration** – changes to the configuration module of ClaimCenter
3. **docs** – contains this design document
4. **repository** – contains JAR libraries required by the JMS integration

Merge .idea folder

Merging the .idea folder is required in order to successfully compile and run the ClaimCenter product from Guidewire Studio.

Copy JAR dependencies to repository directory

Copy all files from the repository folder (from a ZIP accelerator package) to the `[ClaimCenter]\repository` folder, keeping the original folder structure.

Merge configuration

To deploy this reference implementation, merge the contents of the configuration directory in the zip file into the configuration module of your ClaimCenter installation using a third-party merge tool.

ClaimCenter

This section lists all of the files that are included in this reference implementation in the `[configuration]` subfolder, with a brief description of each.

- configuration
    - **pom.xml** – Added dependency on jar files used for JMS integration
    - config
        - **config.xml** – Changes to enable the document assistant
        - extensions
            - typelist
                - **SystemPermissionType.ttx** – Add permission for document approval
                - **DocDlvryStatusType_Ext.tti** – New Document Delivery Status typelist
                - **THBatchStatus_Ext.tti** – New Batch Status typelist
                - **THBulkTransactionType_Ext.tti** – Add typelist for Thunderhead bulk transaction processing types
            - entity
                - **Document.thunderhead.etx** – Add review case information to Document entity
                - **ThunderheadBatch_Ext.eti** – New entity with Batch Status information
                - **ReviewCase_Ext.eti** – New entity with Review Cases
        - Import
            - gen
                - **activity-patterns.csv** – Activity patterns for Document Approval
                - **importfiles.txt** – Automatically import the new activity patterns
        - locale

        - **display_en_US.properties** - Updated to include the accelerator displaykeys on the user interface
    - messaging/**messaging-config.xml** – Added Thunderhead destination to messaging infrastructure
    - plugin/registry/
        - **IDocumentProduction.gwp** – Added Thunderhead type to the Document Production plugin
        - **IDocumentTemplateSource.gwp** – Set Thunderhead as Document Template Source
        - **ThunderheadReply.gwp** – Defined Thunderhead Reply plugin (for receiving batch statuses)
        - **ThunderheadSpringPlugin.gwp** – Defined Thunderhead Spring plugin (for dependency injection)
        - **ThunderheadTransport.gwp** – Defined Thunderhead Transport plugin (for sending batches)
    - rules
        - Assignment
            - DefaultGroupActivityAssignmentRules_dir/**DGA07500DocumentApprovalAssignment.gr** – Assigns the document review activity to a group
            - DefaultGroupActivityAssignmentRules_dir/**DGA07600DocumentRejectionActivity.gr** – Assign the document rejection activity to a group
            - DefaultGroupActivityAssignmentRules_dir/**order.txt**
            - GlobalActivityAssignmentRules_dir/**GAA06500DocumentApprovalActivity.gr** – Assign the approval activities
            - GlobalActivityAssignmentRules_dir/**order.txt**
        - EventMessage
            - **ThunderheadSubmit.gr** – Added rule set for Thunderhead submission
            - ThunderheadSubmit_dir
                - **Document.gr** – Added rule for Document events
                - **order.txt** – Defined rule order
            - **order.txt**
        - Preupdate
            - ActivityPreupdate_dir/**APU01000DocumentApprovalActivityApproved.gr** – Mark the document as Approved.
            - ActivityPreupdate_dir/**APU02000DocumentApprovalActivityRejected.gr** – Mark document as Draft again and create review activity
            - ActivityPreupdate_dir/**order.txt** – Defined rule order
            - **DocumentPreupdate.grs** – Define document preupdate rules
                - DocumentPreupdate_dir/
                    - **DPU01000DocumentApproval.gr** – Checks the document has been submitted for approval
                    - DPU01000DocumentApproval_dir/**DPU01100UserHasApprovalPermission.gr** – Handles documents not requiring permission
                    - DPU01000DocumentApproval_dir/**DPU01200UserDoesntHaveApprovalPermission.gr** – Handles documents requiring permission
                    - DPU01000DocumentApproval_dir/**order.txt** – Defined rule order
                - **order.txt** – Defined rule order
    - servlet
        - **servlets.xml** – Added servlets used for hosting DocumentEditor control.
    - web/pcf

- acc/thunderhead
    - activity/**DocumentApprovalDetailWorksheet.pcf** – Custom worksheet for approving document approval activities
- acc/thunderhead/document/**SubmitDocumentYesNoPopup.pcf** Custom screen to check if the document is ready for approval
- claim/documents/**DocumentDetailsInputSet.pcf** – Ensure the document status is not editable
- claim/documents/**DocumentDetailsPopup.pcf** – Changes to allow for document submission after initial creation
- claim/newdocument/
    - **DocumentEditorPopup.pcf** – Added popup with DocumentEditor ActiveX control
    - **NewDocumentFromTemplateScreen.pcf** – Updated to create Thunderhead documents.
    - **NewTemplateDocumentDV.pcf** – Updated to create Thunderhead documents.
    - **ClaimNewDocumentFromTemplateWorksheet.pcf**
    - **NewClaimNewDocumentFromTemplateWorksheet.pcf**
- document
    - **DocumentCreationInputSet.pcf**
    - **DocumentCreationToolbarButtonSet.pcf**
- shared
    - activity
        - **ActivityDocumentsLV.pcf** – Disable documents edit ability for approval activities
    - documents
        - **ClaimDocumentDetailsDV.default.pcf** – Added Edit Content button
- workspace
    - activity
        - **ActivityDetailForward.pcf** – Condition for document approval
        - newdocument
            - **ClaimNewDocumentFromActivityPopup.pcf**

- deploy
    - thunderhead/dll/
        - ***** – DLL files with DocumentEditor ActiveX control (and related files) to be hosted by application container
- gsrc/gw/acc
    - thunderhead
        - cc/enhancements
            - **ActivityEnhancement.gsx** – Enhancements for activity creation
            - **ActivityPatternEnhancement.gsx** – Enhancements for activity pattern retrieval and comparison
            - **DocumentEnhancement.gsx** – Enhancements for approving documents
        - cc/model
            - **ClaimModel.gx** - GX model containing Claim data graph
            - **THAddressModel.gx**
            - **THCCBusinessObjectModel.gx**
            - **THClaimModel.gx**
            - **THCompanyModel.gx**
            - **THContactModel.gx**
            - **THPersonModel.gx**

        - **THPolicyModel**.gx
        - **THUserContactModel**.gx
    - cc/pcf
        - **NewDocumentFromTemplateScreen.gs** - A backing class to encapsulate code from the New-DocumentFromTemplateScreen.pcf
    - cc/plugin
        - **CCThunderheadDocumentProduction.gs** - ClaimCenter implementation of the Thun-der-headDocumentProduction plugin
    - cc/service/
        - **CCThunderheadService.gs** - ClaimCenter implementation of the ThunderheadService class
    - configuration
        - **ConfigProviderFactory.gs** - A factory for ThunderheadConfigurationProvider
        - **ThunderheadConfig.properties** - Properties file holding all the Thunderhead configu-ration used by the accelerator
    - messaging
        - **ActivityHelper.gs** - Helper class to manage activities
        - **BatchHandler.gs** - Class handling the Thunderhead Batch job status
        - **DocumentHelper.gs** - Helper methods for working with documents sent/received to/from Thun-derhead service
    - web
        - **DocumentEditor.gst** - HTML template with embedded DocumentEditor ActiveX con-trol
        - **DocumentEditorServlet.gs** - Servlet used to render Thunderhead's DocumentEditor ActiveX control
        - **DraftEditor.gst** - HTML page to display DraftEditor javaScript
        - **ThunderheadStoreReviewCaseServlet.gs** - Servlet used to receive Review Case from Docu-mentEditor
- common
    - spring/util
        - **SpringApplicationContext.gs** – Spring Framework configuration file
    - thunderhead/api/
        - wsdl
            - **CMSAPIWebWrapped.wsdl** – WSDL with CMSAPIWebWrapped service defi-nition
            - **JobAPIWebWrapped.wsdl** – WSDL with JobAPIWebWrapped service defini-tion
        - **CMSAPIWebWrapped.wsc** – WSI collection for Thunderhead CMSAPIWebWrapped service
        - **CMSAPIWebWrapperImpl.gs** – Default implementation of the wrapper interface call-ing the WSI client
        - factory
            - **APIFactory.gs** – Factories for creating web services
            - **IAPIFactory.gs** – Interface for API factory
        - **ICMSAPIWebWrapper.gs** – Interface abstracting methods of CMDAPIWebWrapped service (for testability)
        - **IJobAPIWebWrapper.gs** – Interface abstracting methods of JobAPIWebWrapped ser-vice (for testability)
        - **JobAPIWebWrapped.wsc** – WSI collection for Thunderhead JobAPIWebWrapped ser-vice
        - **JobAPIWebWrapperImpl.gs** – Default implementation of the wrapper interface call-ing the WSI client
        - rest
            - **ITHSCRestClient.gs** – Interface for Rest Client

        - **THJobType.gs** – Interface for Thunderhead Job Types
        - **THSCDocumentService.gs** – Thunderhead Smart Communications Restful endpoint for generating drafts, finalizing drafts and other similar functions
        - **THSCRestClient.gs** – Class with methods to create Thunderhead Smart Communications Restful endpoints
        - **THSCSearchService.gs** – Thunderhead Smart Communications Restful endpoint with methods to search for templates and other related function

- thunderhead/auth
    - **UsernameTokenWsdlConfigurationProvider.gs** – Configuration provider used to add authorization to Thunderhead services
- thunderhead/bo
    - **BaseBusinessObject.gs** – Base business object class (to be extended later by product-specific classes)
    - **CCBusinessObject.gs**
- thunderhead/configuration
    - **IConfigProviderFactory.gs** – An interface defining a factory for configuration provider classes
    - **ThunderheadConfigProvider.gs** – Helper class for accessing Thunderhead configuration
- thunderhead/dto
    - ***.gs** – These are all objects used as data transfer objects for marshalling data to Thunderhead Smart Communications
- thunderhead/error
    - **ErrorInfo.gs -** A class used to extract error information from Thunderhead Smart Communications replies
- thunderhead/exception
    - **THSCException.gs** – A class to represent Thunderhead Smart Communications exception information
    - **ThunderheadConfigurationException.gs** – Wrapper class for Thunderhead checked exceptions
    - **ThunderheadRuntimeException.gs** – Wrapper class for thunderhead runtime exceptions
- thunderhead/messaging
    - **BaseReply**.gs – Base class for message reply plugin implementations
    - **BaseTransport.gs** – Base class for message transport plugin implementations
    - **DocumentJmsReply.gs** - Implementation of the MessageReply plugin. Internally it's a listener for events coming from Thunderhead's output JMS queue
    - **DocumentJmsTransport.gs** – Implementation of the MessageTransport plugin, which sends message to the Thunderhead system using its JMS input queue
    - **DocumentRestTransport.gs** – Implementation of the MessageTransport plugin, which sends messages to the Thunderhead system using RESTful web services
    - **DocumentWebReply.gs** – Implementation of the MessageReply plugin. Uses MessageProcessor for periodic processing of the Thunderhead Batches
    - **DocumentWebTransport.gs** – Implementation of the MessageTransport plugin, which sends message to the Thunderhead system using its WebServices
    - **IMessageFinder.gs** – Abstracts the logic for obtaining the list of messages to process
    - **IMessageListener.gs** – Helper interface used by MessageProcessor abstracting the listener
    - **JmsConnectionManager**.gs – Helper class for managing JMS connections
    - **MessageProcessor.gs** - Simple implementation of the message queue processor with one listener
    - **PeriodicActionExecutor.gs** - Helper class for periodic executing a specified action
    - **ThunderheadApiConst.gs** - Helper class with Thunderhead API constants
- thunderhead/plugin

      - **ThunderheadDocumentProduction.gs** - Base class for DocumentProduction plugin implementa-tions
      - **ThunderheadDocumentTemplateDescriptor.gs** - Simple IDocumentTemplateDe-scriptor imple-mentation to represent Thunderhead document templates
      - **ThunderheadDocumentTemplateSource.gs** - Document template source plugin for accessing document templates stored in the Thunderhead
      - **ThunderheadSpringPlugin.gs** - Plugin creating the singleton instance of the SpringApplicationContext
    - thunderhead/service
      - **IThunderheadService.gs** - Interface for the ThunderheadService, the implementation will provide all interaction and logic needed to work with Thunderhead
      - **ThunderheadService.gs** - A façade class for the document and review case rendering provided by Thunderhead
      - **ThunderheadUtil.gs** - Helper class with methods simplifying working with Thunder-head service
    - thunderhead/util
      - **ClassLoaderUtil.gs** - Class with utility methods helping with managing different class loaders (re-quired by client libraries used for JMS implementation)
      - **ThunderheadConstants.gs** - A class to store shared Thunderhead constants
    - thunderhead/xml
      - **ThunderheadXMLCreator.gs** - Abstract class for transaction XML creators. Subclasses are prod-uct-specific
      - **ThunderheadXMLCreatorFactory.gs** - Factory for transaction XML creators
- plugins/shared
  - classes
    - **spring-config.xml** – Spring Framework configuration file
  - lib
    - ***.jar** – Helper JAR files with Spring libraries

Running this Reference Implementation

Configure server

The steps required to deploy and run the reference implementation are relatively simple:

1. Merge the accelerator source code into the product configuration. Be aware that files like **dis-play_en_US.properties** must be merged and not simply overwritten.

2. Update configuration file with Thunderhead environment settings:

   ```
   configuration/gsrc/gw/acc/thunderhead/configuration/ThunderheadConfig.propert
   ies
   ```

   Note: Because this file contains sensitive data like user account names and license keys, it should only be readable by the Guidewire application.

   Sample configuration:

```
#=======================================================
# Common config parameters for Thunderhead integrations
#=======================================================

# Thunderhead batch configuration ID
thunderhead.batch.config.id=154229170
# Thunderhead folder ID
thunderhead.folder.id=157205854
# Thunderhead project ID (recommended default value = 0)
thunderhead.project.id=0

#====================================================================
# Config parameters for Thunderhead Smart Communications integrations
#====================================================================

# Thunderhead Smart Communications host name
thunderhead.sc.hostname=eu3.thunderhead.com
# Thunderhead Smart Communications user name
thunderhead.sc.username=amallou@th-gw8-demo
# Thunderhead Smart Communications API Key EU4 Sales Demo
```

```
thunderhead.sc.key=562a01c7-c454-4edc-8e1c-4ae91cada8c4
# Thunderhead Smart Communications Secret Key EU4 Sales Demo
thunderhead.sc.secret=f7b37b2d-0141-45ec-acb1-7fde312929a3

#===============================================================================================
# Config parameters for Thunderhead NOW integrations (not needed for Smart Communications)
#===============================================================================================

# Thunderhead logon parameters
thunderhead.user=Domain\\User
# WS Timestamp to put into SOAP Envelope (used for authentication)
thunderhead.timestamp.ttl=300
# Interval (in seconds) used to periodically process Thunderhead batch status changes
thunderhead.batch.processing.interval=30
# URL for the Thunderhead web services
thunderhead.http.services.url=http\://localhost\:8086/thunderhead/services
# Thunderhead JMS configuration
thunderhead.jms.jnp.url=jnp\://localhost\:1105
thunderhead.jms.send.queue=queue/thunderhead/JobRequests
thunderhead.jms.response.queue=queue/thunderhead/Responses

#================================
# Config parameters for Guidewire
#================================

# Used as the document production type for thunderhead
# If a document descriptor has this document production type then we know its a
thunderhead document
thunderhead.doc.type=Thunderhead
# Switch to configure which Thunderhead API will be used
# 1 = Thunderhead NOW
# 2 = Thunderhead Smart Communications
thunderhead.api.version=2
# Used for hosting/accessing DLL files (host name)
thunderhead.gw.host=localhost
# Used for hosting/accessing DLL files (port number)
thunderhead.gw.port=8080
```

The following properties can/should be configured:

- Common properties:
    - thunderhead.batch.config.id is the Thunderhead ID of the batch configuration object to use for document production requests from ClaimCenter. See Thunderhead documentation for more information on batch configuration.
    - thunderhead.project.id is an optional property that, if specified, should be the Thunderhead ID of a content project. If this property is specified, Thunderhead will use non-released versions of project items (document templates, batch configurations, business object models, etc.). This feature is useful for template development, where you may not want to follow the complete approval and release workflow in order to test changes, but it should probably be unset (commented out) in production environments.
    - Thunderhead.doc.type is used as the document production type for thunderhead this is used by the platform when deciding which document production plugin to use.
    - Thunderhead.api.version is used to specify which API version is being used, Thunderhead Smart Communications or Thunderhead Now. This will configure the UI and spring injection automatically.
- For Thunderhead Smart Communications:
    - thunderhead.sc.hostname is the name of the Thunderhead Smart Communications host machine
    - thunderhead.sc.username is the username used to log into the thunderhead RESTful services
    - thunderhead.sc.key the API key to be used to connect to thunderhead RESTful services, this API key must belong to the user above
    - thunderhead.sc.secret the secret key to be used to connect to thunderhead RESTful services.

3. Configure the Thunderhead endpoints:
    a. For Thunderhead Smart Communications (cloud) with RESTful services

        i. In the Thunderhead messaging destination configuration set the Transport plugin to ThunderheadTransport. There is no rely plugin for Thunderhead Smart Communications as it is asynchronous.

        ii. The ThunderheadTransport plugin configuration should have a Gosu class of gw.acc.thunderhead.messaging.StatementInvoiceRestTransport

4. Start the server.

Client configuration for Thunderhead Smart Communications

This reference implementation makes use of the JavaScript Draft Editor for Thunderhead Smart Communications

To enable the Javascript Draft Editor for Thunderhead Smart Communications ensure that all Thunderhead Smart Communications properties are set, and then ensure the thunderhead.api.version property is set to 2.

Client browsers must have Javascript enabled. Consult Thunderhead support and the Guidewire Platform Matrix for supported client browser information.

Note: Clients using Internet Explorer should make sure Compatibility Mode is disabled to use the Draft Editor.