

Navigating Compliance in Regulated Industries with CALM



Table of Contents

Who Should Read This White Paper	1
Executive Summary	1
Compliance FAQs	3
Introduction to CALM	5
Use of LLMs in CALM	8
Additional Levers to Control CALM	13
Evaluation and Monitoring	14
Analytics and Continuous Improvement	14
Biases and Limitations	14
Further Reading	15

Who Should Read This Whitepaper

This white paper facilitates and expedites alignment between product and compliance teams in leading enterprises, especially those operating in regulated industries. It covers important considerations when using Rasa’s Conversational AI with Language Models (CALM) in a production environment.

Distinctly different from popular methods, CALM uses Large Language Models (LLM) with a unique approach and can be configured to guarantee zero [AI hallucinations](#). This white paper contains frequently asked compliance-related questions and briefly describes CALM.

“

*“Before **CALM**, I thought there was no way we’d be allowed to use LLMs in our production chatbots. Now that I understand it, the only limitation is getting GPUs provisioned.”*

— Managing Director at a top-3 US bank —

Executive Summary

CALM is an approach developed by Rasa for building next-generation Conversational AI, including chat and voice-based virtual assistants, specifically designed for use in large enterprises. It uses LLMs to retain the control, safety, and security of traditional, “NLU-based” approaches to chat and voice bots.

Specifically:

- By default, assistants developed with CALM only send human-authored responses to end users, guaranteeing zero hallucination. Developers of CALM assistants can choose when and where to enable controlled use of generated language.
- CALM leverages LLMs to understand end-user statements and translate them into commands that align with your custom business logic. For example, a user’s statement like “I want to send money” can be turned into a command such as ‘StartFlow(transfer_money)’. This method simplifies root cause analysis and application debugging. If a conversation goes awry, this framework allows you to precisely identify what went wrong and where. Moreover, you can explain the reasoning behind every response after each conversation.
- CALM uses LLMs to understand users but does not use them to follow business logic. Business logic in CALM is executed deterministically with transparent and customizable code.
- CALM is language-model agnostic and can be used with various LLMs. The choice of LLM is at the user’s discretion, offering flexibility to change models as the technology evolves.
- Support for additional OOTB language models to reduce cost and latency. From a compliance perspective, this means:

- No data ever leaves the enterprise's own environment.
- The model powering CALM can be fully vetted by a model risk management team.
- The enterprise fully controls the model, which does not change behavior until you upgrade to a newer version.

Compliance FAQs

Note: These questions also cover relevant parts of the [OWASP top 10 for LLMs](#).

1. Please describe how the solution addresses the risk of LLM hallucinations.
 - a. CALM assistants only respond with [templated messages](#) unless otherwise [specified](#) by the developer.
2. Please describe how the solution addresses the risk of producing and distributing harmful AI-generated content.
 - a. CALM assistants only respond with [templated messages](#) unless otherwise [specified](#) by the developer.
3. Please describe how the solution addresses the risk of prompt injection attacks allowing a malicious user to override the application's intended behavior.
 - a. All LLM output is discarded if it lacks a [valid set of CALM commands](#).
 - b. The LLM output represents what the end user wants to do to progress the conversation. LLMs do not reason over business logic or processes or decide the next best action. That is handled by the [deterministic FlowPolicy](#).

4. Please describe how the solution addresses the risk of exploiting insecure LLM plugins for XSS, CSRF, SSRF, privilege escalation, or remote code execution.
 - a. CALM does not support third-party plugins.
 - b. CALM assistants can make use of APIs / RPA / RPCs. They do so when your assistant's [business logic](#) reaches an action step. There is no scope for the LLM itself to invoke a function call.

5. Please provide details on the training data used to create any models used in the solution.
 - a. CALM is compatible with [any LLM](#), including proprietary ones you developed. The choice of LLM and the training data used for these models is entirely up to you.
 - b. Guidelines to fine-tune an LLM on-premise/private cloud to use as a command generator.

6. Please describe how the solution mitigates the risk of updates to models behind an API, which could introduce regressions or behavior changes.
 - a. The choice of LLM is [up to the developer](#) using CALM, including the possibility of self-hosting or using a vendor that provides guaranteed static models.

7. Please describe how the solution safeguards PII and other sensitive data.
 - a. PII can be removed from data in transit and at rest by using Rasa's [PII Management](#) feature.

8. Please describe how the solution prevents the excessive agency of an LLM from introducing unwanted behavior.
 - a. In CALM, the LLM output represents what the end user wants to do to progress the conversation. LLMs do not reason over business logic or processes or decide the next best action. That is handled by the [deterministic FlowPolicy](#).

- b. CALM assistants can make use of APIs / RPA / RPCs. They do so when your assistant's [business logic](#) reaches an action step. There is no scope for the LLM itself to invoke a function call.
9. Please explain how the solution addresses the explainability of model predictions.
 - a. Every dialogue turn in a CALM assistant is associated with a set of commands describing what the end user is trying to achieve. If an LLM generates an incorrect prediction (e.g., the wrong commands were issued), it is straightforward to identify and understand what the correct prediction should have been. This clarity also simplifies the process of identifying similar issues in [other conversations](#).
10. Please describe how the solution can be monitored and its quality assessed.
 - a. Rasa provides functionality for [end-to-end testing](#) of CALM assistants to guard against regressions.
 - b. Rasa provides an [ETL pipeline](#) for integrating conversation events into a data warehouse and monitoring performance.

Introduction to CALM

Conversational AI with Language Models (CALM) is an LLM-native approach to building chat and voice-based virtual assistants. It comprises three core elements: Business Logic, Dialogue Understanding, and Conversation Repair. Of these core elements, only the Dialogue Understanding component uses LLMs. These components have the following responsibilities:



Business Logic describes the steps required to complete a user’s task. For example, transferring money to another account. Tasks are defined in a format called **flows** that describe (i) what information is needed from the user (e.g., the amount of money and the recipient), (ii) what information is needed from APIs (e.g., the user’s account balance), and (iii) any branching logic based on the information collected. An example of a flow definition can be found [here](#).

The business logic is specified by the AI assistant developer and executed deterministically by the [Flow Policy](#), a logical engine. The Flow Policy proceeds to the next step in the flow only when the previous steps have been completed and all conditions have been met.

The flow definition includes identifiers referencing the **templated messages** the assistant will send to the end user (e.g., `utter_ask_amount`).

Dialogue Understanding translates between what end users say to the assistant and your business logic. Dialogue Understanding is performed by the `LLMCommandGenerator` component. As suggested by the name, this component leverages the in-context learning abilities of LLMs; however, it does not produce arbitrary text. The Dialogue Understanding component output is a short sequence of commands (typically 1-3) describing how the end user wants to progress the conversation with the assistant. Rasa parsed and interpreted these commands to progress through the business logic. The following commands are allowed:

CALM Commands

```
StartFlow(flow_name)
```

```
SetSlot(slot_name, slot_value)
```

```
CorrectSlot(slot_name, slot_value)
```

```
Clarify(flow_name_1, flow_name_2, ...)
```

```
ChitChat
```

```
KnowledgeAnswer
```

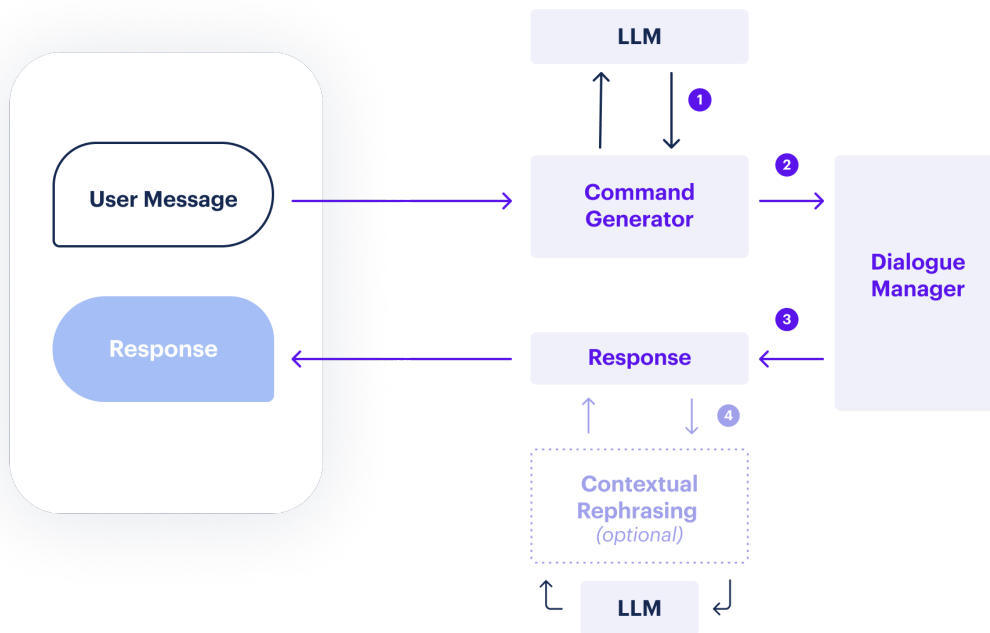
```
HumanHandoff
```

```
Error
```

Conversation Repair defines a set of default flows (called patterns) that describe how your assistant behaves in conversations that deviate from the “happy path” of a flow. For example, an end user may change their mind, switch context, correct something they said earlier, or say something that requires further clarification. These situations are detected by the Dialogue Understanding component, and the corresponding commands (e.g., `CorrectSlot`, `Clarify`) are produced as output. These commands are processed by the Flow Policy, which follows the logic defined in the patterns. Therefore, the process remains consistent with conversations that follow the happy path.

When an end user sends a message to a CALM assistant, the following takes place:

1. The Command Generator component translates the conversation so far (including the latest user message) into a set of commands. The output from the LLM is validated using regular expressions to ensure it complies with the command grammar.
2. These commands are processed by the Dialogue Manager and used to deterministically progress the conversation according to the business logic.
3. The assistant's next action, including a templated, human-crafted response to send back to the end user, is selected.
4. **Optionally**, the templated response is [rephrased](#) to account for context and increase fluency.



Use of LLMs in CALM

Multiple components of Rasa can use LLMs. The only component required to build a CALM assistant is Dialogue Understanding. Two optional but frequently used LLM-powered components are Contextual Rephrasing and Enterprise Search, described below. For any of these components, the CALM assistant developer fully determines the choice of LLM and any configuration parameters.

Dialogue Understanding

As mentioned above, the Dialogue Understanding is the responsibility of the LLMCommandGenerator. This component uses the in-context learning abilities of LLMs, producing a sequence of commands as its output. From a governance and security perspective, the implications of this approach include:

- The command generator can only produce a sequence of allowed commands, not arbitrary text
- The command generator output is not exposed to end users
- Invalid output from the command generator is discarded and ignored

This table provides examples of end-user messages and their translation into commands:

User Message	Command Output
I want to transfer money	StartFlow(transfer_money)
I want to transfer \$55 to John	StartFlow(transfer_money), SetSlot(recipient, John), SetSlot(amount, 55)
Actually I meant \$45	CorrectSlot(amount, 45)

Contextual Rephrasing

The Contextual Response Rephraser is an **optional** component that can be used in conjunction with CALM to improve the end-user experience.

When invoked, the Contextual Response Rephraser works as follows:







- The AI assistant selects the next action for the assistant to execute along with the corresponding templated response.
- An LLM is prompted with the transcript of the conversation so far and is tasked with rephrasing the assistant's latest response to make it more fluent and contextual.

By default, the Contextual Response Rephraser is **deactivated**, meaning your assistant will never send generated text to end-users. Developers can activate rephrasing for [individual templated responses](#).

From a governance and security perspective, the implications of this approach include:

- Because the LLM is used to rephrase an existing message while preserving its meaning, the developer determines the assistant's response intention. The developer decides what the assistant should say, and an LLM allows some freedom in *how it is said*.
- Because the starting point is a templated message with a known identifier, it is always known what message was sent to each user. If any issues arise from generated text, it is straightforward to identify all conversations where a similar issue might have occurred.
- The Contextual Response Rephraser is disabled by default. The AI assistant developer can enable it on a per-response basis. For example, using rephrasing for “small talk” dialogue while keeping templated responses for transactional conversations.
- The developer has full control over customizing the Contextual Response Rephraser by modifying the prompt and the LLM configuration parameters (e.g., temperature) to adjust the diversity level.

This table provides examples of conversations with and without contextual rephrasing:

 I want to add my partner to my credit card.	
Templated Message	With Rephrasing
 Sorry, I am unable fulfill that request.	 Unfortunately, I'm not able to add additional users to your credit card.
 application	
Templated Message	With Rephrasing
 Which of these would you like to do: * start_application * check_application_status	 Would you like to create a new application, or are you asking about the status of an existing application?

The following is an example prompt template used for contextual response rephrasing:

The following is a conversation with an AI assistant. The assistant is helpful, creative, clever, and very friendly.

Rephrase the suggest AI response staying close to the original message and retaining its meaning. Use simple english.

Context / previous conversation with the user:
{{history}}

{{current_input}}

Suggested AI Response: {{suggested_response}}

Rephrased AI Response:

Enterprise Search and Retrieval-Augmented Generation (RAG)

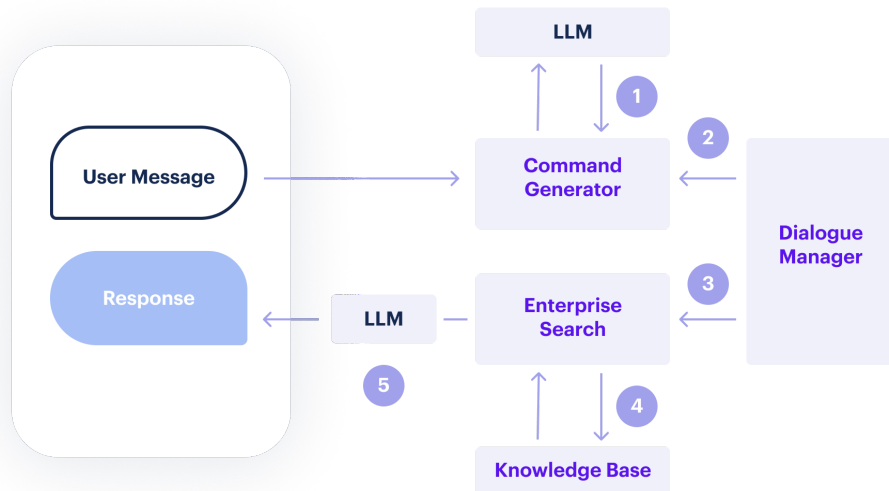
Enterprise Search is an **optional** component of CALM that leverages Retrieval-Augmented Generation (RAG) to handle informational dialogue. Informational Dialogue refers to end-user queries that can be answered based on some documentation and does not require read/write access to real-time data via APIs.

The RAG approach has become popular because of its short time to value. An end user's question is used to perform information retrieval on a collection of relevant documents, producing a collection of potentially relevant document snippets. An LLM is then used to answer the user's question based on the information contained in the retrieved snippets. This avoids manually crafting responses to every user question, significantly reducing the overall implementation effort.

If enterprise search is enabled and an end-user asks a knowledge-related question, the following takes place: [Steps 1-3 are identical to the [previous case](#)]

1. The Command Generator component translates the conversation so far (including the latest user message) into a set of commands. The output from the LLM is validated using regular expressions to ensure it complies with the command grammar.

2. These commands are processed by the Dialogue Manager and used to deterministically progress the conversation according to the business logic.
3. The assistant's next action, including a templated, human-crafted response to send back to the end user, is selected.
4. The Enterprise Search Policy is invoked in response to the Dialogue Understanding component yielding a KnowledgeAnswer command. This queries a knowledge base for relevant content.
5. The knowledge query results and the user's question are combined into a prompt for the LLM, which generates an answer. Developers can fully customize this prompt template.



The following is an example prompt template used for generating the answer in step 5:

Given the following information, please provide an answer based on the provided documents and the context of the recent conversation.

If the answer is not known or cannot be determined from the provided documents or context, please state that you do not know to the user.

Relevant Documents

Use the following documents to answer the question:

```
{{ documents retrieved via enterprise search }}
```

```
### Current Conversation
```

Transcript of the current conversation, use it to determine the context of the question:

```
{{ current conversation }}
```

Based on the above, please formulate an answer to the question or request in the user's last message.

It is important that you ensure the answer is grounded in the provided documents and conversation context.

Avoid speculating or making assumptions beyond the given information.

Your answer:

From a governance and security perspective, the implications of this approach include:

- Enterprise Search is **deactivated** by default.
- Enterprise Search uses a RAG approach; thus, LLM-generated answers are sent to your end users. This introduces risks like hallucination, prompt injection, and more (see the [OWASP top 10 for LLMs](#)). You may consider investing in additional tooling to guard against these risks.

Additional Levers to Control CALM

One of the most important jobs of the Dialogue Understanding component in CALM is deciding when to issue a StartFlow command to initiate a business process/task for the user. As a result, Rasa provides additional ways to control when flows are started.

Flow Conditions

A flow definition can contain one or more conditions. If these conditions are not met, the flow cannot start. Common use cases include restricting certain flows to specific user segments

and activating/deactivating flows dynamically based on external systems. See the [documentation](#).

NLU Triggers

Conversely, NLU triggers allow you to determine when to activate a specific flow, disregarding any LLM output. Common use cases include subjects where a user needs to be immediately connected with a human agent or where regulatory rules specify the necessary answer. See the [documentation](#).

Evaluation and Monitoring

Rasa includes a feature for conducting end-to-end testing of AI assistants. End-to-end tests are useful tools for evaluating AI assistants and guarding against regressions.

An end-to-end test in Rasa documents a conversation between an end user and an AI assistant. It captures how the assistant should respond at every turn and any expected side effects.

A diverse and comprehensive suite of end-to-end tests is valuable for evaluating a CALM assistant.

Analytics and Continuous Improvement

For each conversation handled by a CALM assistant, Rasa provides rich information that can be used for debugging, root cause analysis, and analyzing trends. This information is made available through the [analytics ETL data pipeline](#), enabling its transfer to internal BI systems for analysis and reporting.

The following is available for each conversation turn:

- The user message
- The commands generated by the Dialogue Understanding component

- Any active flows and the status of each flow
- The values of any slots
- The identifiers of each response sent by the assistant

Biases and Limitations

LLMs are trained on extensive internet data, inheriting innate biases present in that data. CALM uses LLMs to understand users, not to reason about a business process or make decisions. As such, the main risk of bias affecting users is potential misunderstandings.

This is particularly relevant for speakers of dialects, minority languages, etc. In voice applications, the likelihood of errors in automatic speech recognition is higher for these groups. While many LLMs are multilingual, there is no guarantee LLMs perform consistently across all languages. Therefore, enterprises offering an AI assistant in multiple languages must rigorously test to ensure a high-quality user experience across different languages and geographies.

Further Reading

Rasa's [documentation](#) provides more detail on usage and inner workings.

Rasa's [YouTube channel](#) provides in-depth algorithm explanations, how-to guides, webinars, and more.



Interested in learning more about CALM?

Visit us at rasa.com/connect-with-rasa/ for more information.