~~Chat Solution Technical Specifications~~

~~Version 3.4~~

~~December 25, 2018~~

Deprecated - Moved to [Hybrid Chat Confluence site]

~~In this document Chat solution components, its architecture and integration interfaces are explained. It's a technical document explaining the architecture, communication flows, API driven integration interfaces and more.~~

~~Abbreviations~~

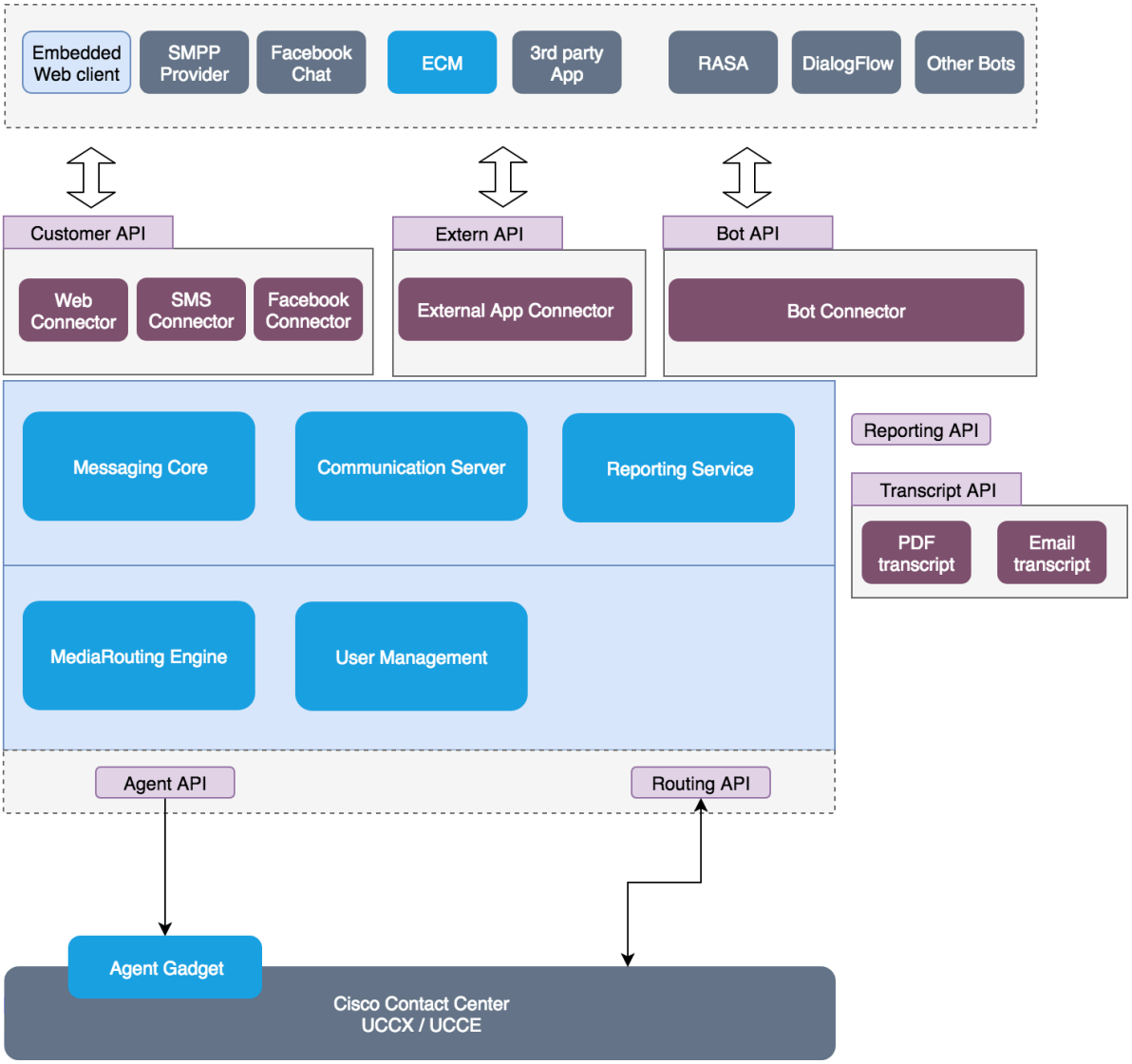
~~Short Name~~	~~Full Name~~
~~MRE~~	~~Media Routing Engine~~
~~Gadget~~	~~Agent Gadget~~
~~Finesse~~	~~Cisco Finesse~~
~~Server~~	~~EF Chat Solution~~
~~CIM~~	~~Customer Interaction Manager~~
~~ECM~~	~~Expertflow Campaign Manager~~
~~UMM~~	~~User Management Module~~
~~MRD~~	~~Media Routing Domain~~

Related Documents

Chat Solution API Developer Guide	This guide covers integration interfaces of the chat solution for customer chat channel, any bot, any agent interface

Chat Solution Component Architecture

The Chat solution is structured in the following layers.



Core Components

Core solution core components are covered in the middle tier with a light-blue background.

Messaging Core	A message routing logic is defined in the Messaging core. It routes messages between customer channel, bot, and the human agent. It keeps track of a conversation and its participants. It communicates with: - customer channels via Customer API - Bots via Bot API - Human agents via Agent API

Communication Service	Communication Server maintains updated agent states for non-voice MRDs. It routes messages to available chat agents. It processes all the agent state change events.
Reporting Service	A real-time data publishing service that publishes all the chat solution reporting events to Chat solution database for the reporting purposes.
Media Routing Engine	Used to reserve agents and route new messages. The Routing API running on top of this service integrates with Media Routing Engine (MRE) to synchronize fetch and reserve agents for media-blending with Cisco contact center agents.
User Management	handles both Standalone agents and contact center agents. It stays in sync with the Cisco contact center via Synchronizer.

Customer Gadget

Chat transcript - an email is sent to the customer's email specified. Chat solution uses SMTP configurations to send email.
Only SMTP configurations are supported.

Agent Gadget

Agent chat gadget is the web application interface for contact center agents to receive and answer chat requests. The gadget may be loaded in Finesse as a finesse gadget. See Agent gadget user guide for the gadget features and limitations.

Chat solution APIs

The Chat solution core communicates via the published API interfaces with the external components.

Customer API	Integrate any customer channel such as Facebook, web-site, a mobile app or similar for embedded chat with the corporate branding and other styling requirements.
Extern API	For 3rd party application integration for 3rd party application to customer channel messaging. ECM uses this interface for sending campaign SMS messages to the customer phone number.
Agent API	An interface for connecting a human agent. The system allows 3rd party agent channel management through this interface
Bot API	For connecting any 3rd party chatbot on the defined list of API interfaces
Routing API	Used to reserve agents and route new messages. The Routing API running on top of this service integrates with Media Routing Engine (MRE) to synchronize fetch and reserve agents for media-blending with Cisco contact center agents.

Transcript Service	Store chat transcripts at the end of a chat session.
Email Transcript	Sends customer the chat transcript via the email address provided by the customer
PDF Transcript	This transcript service implementation make the PDF transcript available for the customer to download at the end of a chat session.

API Connectors

API connectors are the client applications that connect with the Chat solution over the published communication interface. Following are the out-of-the-box connectors available with the Chat solution.

Web Connector	Use Customer API to connect an embeddable HTML5 web-widget in a web-site, a web or a mobile app.
SMS Connector	Runs on the Customer API and connects with 3rd party SMS gateways over an SMPP. REST, or custom APIs for sending and receiving SMS messages
Facebook Connector	This Customer API implementation enables facebook channel with the Chat solution.
Bot Connector	Allows integration with 3rd party bots such as IBM Watson, DialogFlow, and more on the published Bot API

To develop your own connection application on the available integration interfaces, see Chat Solution API Guide.

Media Routing Features & Definitions

The Media Routing Engine supports agent-state per MRD and task-state per MRD operations. Agent selection is done based on precision queue configurations. It maintains routing related configuration (Precision queues, contact center/standalone agents, agent selection attributes, MRDs) in its internal datastore for persistence.

Media Routing Domains (MRD)

MRD organizes how requests for each communications medium are routed to agents. An MRD for each media channel is configured at the deployment time.

Types of MRDs

➔ Interruptible/Non-Interruptible MRDs
➔ Interactive/Non-Interactive MRDs

Interactive/Non-Interactive MRDs

Interactive MRDs maintain a session b/w customer and agent, in which both communicate in real time. For example a web chat session.
Non-interactive MRDs does not maintain a session, e.g., an email message or an SMS. These tasks are *asynchronous* as customer can send an email, close window and check back later for a response.

Features and their values/actions difference in case of Interactive/non-interactive MRDS

Feature	**Interactive Task/MRD Suggested Value/Action**	**Non-Interactive Task/MRD suggested value/Action**
Requeue on Recovery - System restores from failure	**False** - customers are waiting at an interface for an agent and can be notified if there is a problem. we don't need to requeue these tasks	**True** - customers are not waiting at an interface for an agent, and there is no way to alert them that there was a problem. We need to requeue these tasks.

Max Concurrent Tasks per agent per MRD

The limit is configurable. Max Concurrent Tasks Limit can be configured from Media Routing Engine Administration.

Interruptible and Non-Interruptible MRDs

- **Interruptible:** Agents handling tasks in the MRD can be interrupted by tasks from other MRDs. Non-interactive MRDs, such as an email MRD, are typically interruptible.
- **Non-Interruptible:** Agents handling tasks in the MRD cannot be interrupted by tasks from other MRDs. The agents can be assigned tasks in the same MRD, up to their maximum task limits. For example, an agent can handle up to three non-interruptible chat tasks; if the agent is currently handling two chat tasks, the media routing engine can assign the agent another chat, but cannot interrupt the agent with an email. Interactive MRDs, such as a chat MRD, are typically non-interruptible. Voice is non-interruptible.
- **Mixed**: agent's active chat sessions will remain active while he'll continue to receive an inbound call as well because MRE will not change the agent's Finesse state.

Handling interrupts

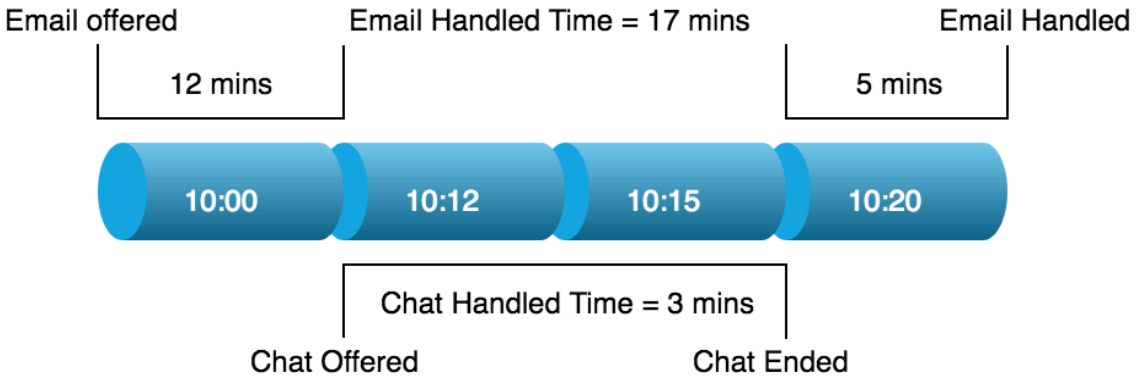
Following approaches are used to handle interrupts:

- **Accept Interrupts**
  - ➔ When an agent is interrupted by a task from a non-interruptible MRD while working on a task in an interruptible MRD, interrupt event is accepted.
  - ➔ The agent state & task dialog state in the interrupted MRD change to INTERRUPTED.

➔ The agent cannot perform dialog actions while a task is interrupted.

The agent's time on task stops while the agent is interrupted.

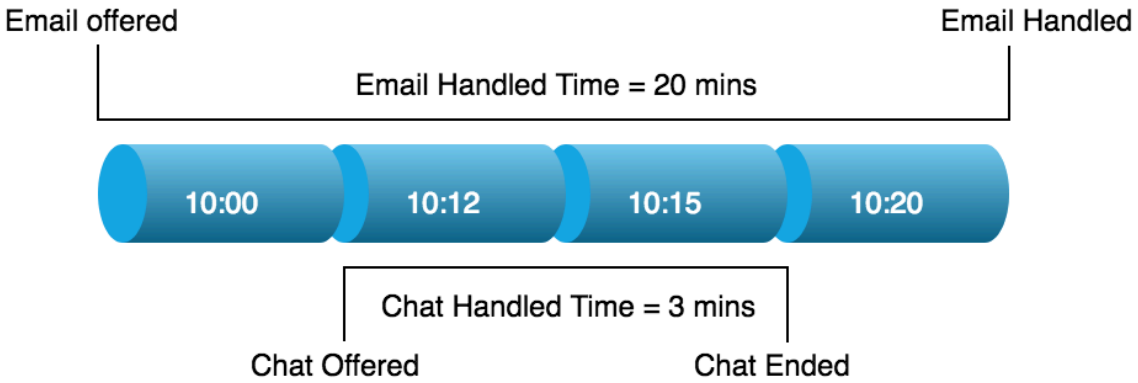
Example: An agent has an email task for 20 minutes, and is interrupted for 3 of those minutes with a chat task. The handled time for the email task is 17 minutes, and the handled time for the chat task is 3 minutes.



- **Ignore Interrupts**
  - ➔ When an agent is interrupted by another task while working on a task in an interruptible MRD, interrupt event is ignored.
  - ➔ The new task does not affect any of the agent's other assigned tasks. The agent state & task dialog state in the interrupted MRD does not change.

The agent can perform dialog actions on the original task and the interrupting task at the same time. The agent's time on the original task does not stop while the agent is handling the interrupting task.

Example: An agent has an email task for 20 minutes, and is interrupted for 3 of those minutes with a chat task. The handled time for the email task is 20 minutes, and the handled time for the chat task is 3 minutes. This means that during a 20-minute interval, the agent handled tasks for 23 minutes.



Tasks Routing to Agent

It depends upon 2 things:

1. **Agent Mode:** If agent's mode is routable then the task is routed to agent else it is not
2. **Agent State:** Agent's state in MRD. Following are agent states

a. Ready
b. Reserved/Active/Work-Ready/Interrupted/Busy
c. Not-Ready

* Every MRD has its *own state*

Conditions for a task to be routed

The task is routed to an agent if ALL these conditions are TRUE:

1. Agent's mode is routable, AND
2. An agent is in a state other than Not-Ready, AND
3. The agent has not reached Max task limit in MRD, AND
4. The agent is NOT working on a task in a different Non-Interruptible MRD.

Non-Routable Mode; Why?

The non-routable mode is helpful in these kinds of scenarios:

- ➔ The agent is in Ready state and working on tasks
- ➔ It's almost off time now, but all tasks are not done so far. The agent needs to complete these tasks ensuring new tasks will not be routed to him
- ➔ For this, switching to the Not-Ready state during active tasks will show tasks ended on the time of switching to Not-Ready in reports (while the actual duration of a task in longer as tasks are not ended on agent's desktop upon switching state from Ready -> Not Ready)
- ➔ An agent can set himself Non-Routable in this case and no new task will be routed to him. He can complete his tasks while staying in Ready state.

Chat Solution Components Interaction

Alternative solution, 14 Jan 2019, Andreas

See also here. We route individual messages, not chat sessions! The chat engine determines

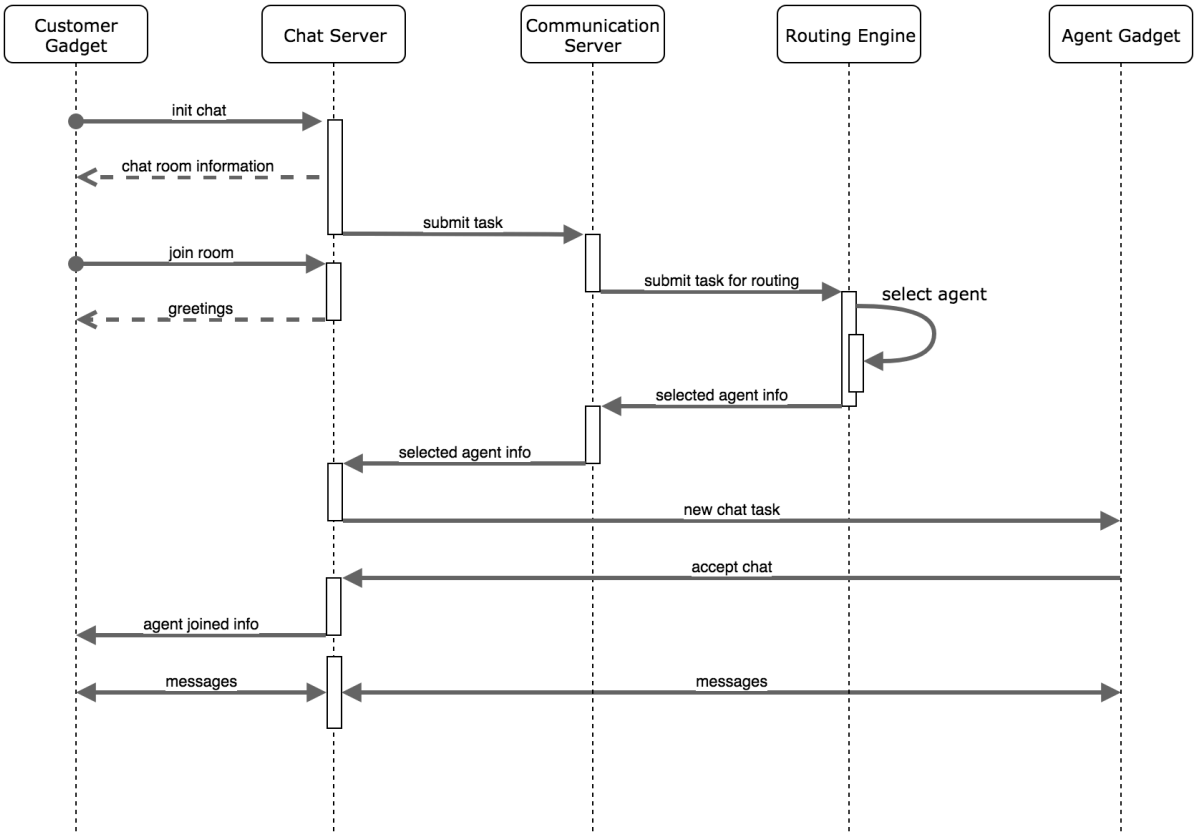
1. Which chat has which work volume (depending on channel intensity, dialog state, unserved intents)
2. Which agent currently has which total work volume on his roster
3. How much work volume to give to each agent
4. Whether a new full-time agent is requested or not
5. Which chat sessions should be dropped from a chat roster (timeout or explicit termination messages)

New Chat arrives, is processed by bot with suggested answer, answer is not yet sent out.
Hybridchat (HC) decides whether an agent is required for this confidence score or not.
Create a CIM activity to store the original message, along with bot information. IF agent required, add the tag “human required”, if not required, send back bot-suggested answer.

If agent is required, determine whether whether an agent is currently subscribed to activities of that Customer. If not, determine based on concurrent traffic load per agent and (if provided) expected work volume of that chat which agent should next take that chat, and forward it to that agent’s roster.
If the chat volume per agent is more than a full-time work volume, query Cisco UQAPI (or our Routing Engine) for an additional agent.
Once the agent sent an answer, start a timer of 10 minutes. After 10 minutes, unsubscribe the agent from activities of that customer, and have the chat disappear from the roster of that agent.

New Chat (non-Hybrid mode)

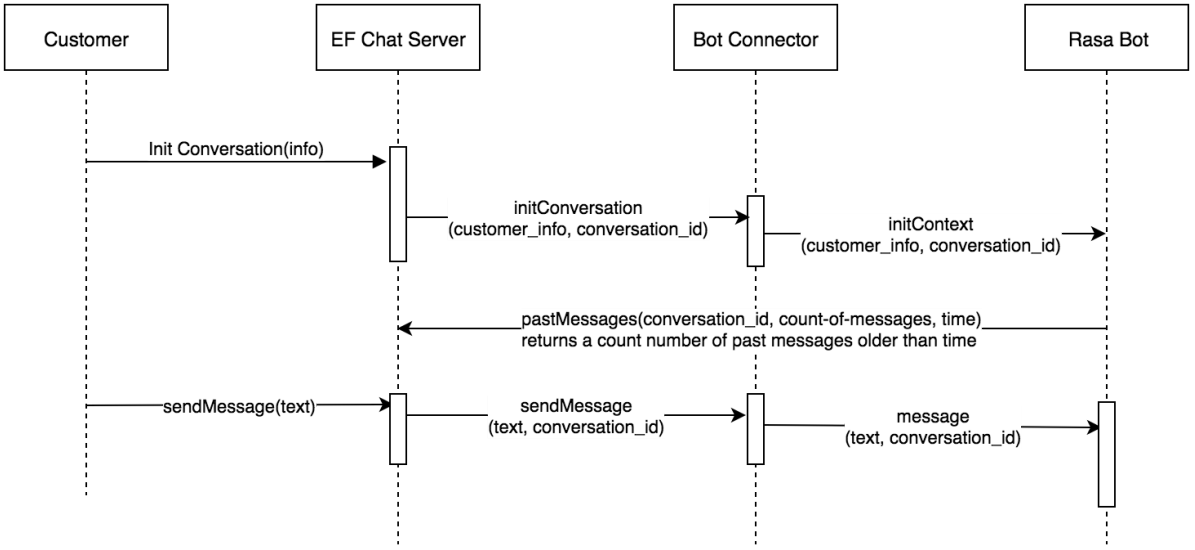
A new customer chat request is routed to an available agent.



a. The customer initiates chat from Customer Gadget. Customer Gadget sends init chat request via HTTP/S REST
b. Chat Server sends room information back to Customer Gadget
c. Chat Server submits a task to Communication Server
d. Communication Server submits new task request to Routing Engine
e. Routing Engine selects agent and sends agent and task information back to Communication Server
f. Communication Server sends agent information to Chat Server
g. Chat Server sends a new chat request to the selected agent
h. The agent accepts a chat from Agent Gadget. Agent Gadget sends acceptance to Chat Server via HTTP/S REST
i. Chat Server forwards agent info to Customer Gadget
j. Chat Server enables messaging between Customer and Agent

Initialize Bot Context with Past Messages on New Chat

Chat server intimates the connected Bot upon a new chat request. The Bot may load the recent past messages to initialize the bot context to better answer new customer request messages.
The Bot calls Past Messages API to load recent past messages at the start of a conversation.



Chat Solution APIs

Chat Solution API Developer Guide states API calls and example workflows for development on respective APIs.

Conversation = CIM Interaction

A conversation (=CIM Interaction) identifies a set of exchanged messages (= CIM Activity) between chat participants. Every conversation can have one or more participants of the following type:

a) Customer
b) Agent
c) Bot

A web chat session has a different conversation for every new chat.
An SMS conversation with a customer remains in the same conversation object throughout the lifetime of a customer.
A Facebook chat with a customer is tracked via the same conversation object throughout the lifetime of a customer.

What-if Scenarios

[OpenShift] If Chat server is down, all session will be disconnected.	If a dependent component such as UMM is restarted, Hybrid Chat takes around 10 minutes to resume its services.