Telegram Bot Intent Classifier — comms-responder

Telegram Bot Intent Classifier — comms-responder

MC: #99290, #99331 **Status:** Live **Last deploy:** 2026-05-05 22:47 UTC **Code SHA-256:** `233baff845b6c16153f900d1cab9756f84a72999f6425e375830a343b4870d36`

Related runbook: SQLite DB Backup — Pillar #9 LITE (#99248)

---

1. Overview

The **comms-responder** is the intent-routing brain of the ALAI Telegram bot. It receives every inbound CEO message and classifies it into one of five intent categories before generating a response. Prior to MC #99290, the bot had a task-creation bias — informal messages such as "Bok" or "Kako si?" triggered unsolicited offers to create MC tasks. The fix is **prompt-only**: no code was changed in `comms-responder.js` (802 lines, unchanged) or `telegram-agent.js`.

Root cause (from audit.md): The original `main-system-prompt.md` placed the Task Actions section (lines 20-45) before conversational rules and had an over-broad MAYBE branch — "If unsure whether it is actionable, ASK" — which caused haiku-model responses to offer task creation for purely casual messages, since 20 live MC tasks were injected unconditionally into every context window.

Fix applied 2026-05-05:

- Explicit 5-category intent classifier added at the TOP of the prompt (before context/actions).
- Task Actions section reordered to end and renamed to "ONLY for task-create intent".
- Aggressive MAYBE/ASK fallback removed.
- Conversational mode made the explicit default.

MC #99331 added code-level gate (2026-05-05): CEO live test "Bok" still triggered "faza 2.5" hallucinations from MC task titles. While #99290 added intent classification to the PROMPT, `comms-responder.js` still injected `mc.js list` output unconditionally into every context window. #99331 added a CODE classifier that gates the injection: only `task-create` and `status-query` intents now receive the MC task list. This forms a two-layer defense (prompt + code) against context drift.

Key files:

- Prompt: `/Users/makinja/system/prompts/extracted/comms-responder/main-system-prompt.md`
- Bot logic: `/Users/makinja/system/tools/comms-responder.js` (802 lines; #99331 added `classifyIntent()` function)
- VM daemon: `alai-telegram-agent.service` on Azure VM (4.223.110.181)

---

2. Intent Categories

Every inbound message is classified into exactly one of these five categories before a response is generated. Classification is silent (not shown to the user).

Category	Trigger signals	Response mode	Task action?
`greeting`	Bok, Selam, Zdravo, Hey, Ciao, opener phrases	Short friendly reply. No task mention.	Never
`chitchat`	Kako si?, Šta radiš?, casual conversation	1-2 sentences conversational. No task offer.	Never
`status-query`	Da li je X gotov?, Šta ima novo?, Koji je status...?	Pull from task context, answer directly and concisely.	Never
`question`	Kako radi X?, Zašto je Y?, factual or technical ask	Answer from context. No task offer unless explicitly asked.	Never
`task-create`	Kreiraj task, Napravi MC za..., Dodaj task, Otvori MC — explicit verb + task directive	Create task, confirm with ID.	ONLY this category

Default rule (prompt line 17): If category is not clearly `task-create`, respond conversationally. NEVER offer to create a task unprompted.

---

3. How It Works

The classification logic lives entirely in the system prompt file:
`/Users/makinja/system/prompts/extracted/comms-responder/main-system-prompt.md`

The prompt is loaded at runtime by `comms-responder.js` via the `buildSystemPrompt()` function. No intent classification code exists in the JS layer — the LLM (haiku to sonnet chain) performs STEP 1 classification silently before generating a response.

Request flow:

1. CEO sends Telegram message.
2. `alai-telegram-agent.service` (VM) receives update via Telegram Bot API webhook.
3. `telegram-agent.js` calls `comms-responder.js getResponse(message, history)`.
4. `comms-responder.js` builds system prompt from `main-system-prompt.md`, injects scope block + task context + today's date.
5. LLM classifies intent (STEP 1) silently, then generates response.
6. If intent = `task-create`: response includes `json:action` block. `telegram-agent.js parseAndExecuteActions()` extracts and runs it against `mc.js`.
7. For all other intents: plain conversational response, no action block.

Note on MC task injection: `comms-responder.js` lines 191-207 load 20 open MC tasks unconditionally into every context window (to enable `status-query` answers). This is by design; the classifier gate prevents this context from biasing non-status responses toward task creation.

3a. Code-Level Intent Gate (#99331)

While #99290 added prompt-based intent classification, CEO live testing on 2026-05-05 revealed that the greeting "Bok" still triggered hallucinations like "faza 2.5" from MC task titles injected into the context window. The root cause: `comms-responder.js` still executed `mc.js list --limit 20` unconditionally for **every** message, regardless of intent.

MC #99331 added a CODE classifier (regex-based, 5 categories, lines 185-219) that runs **before** building the system prompt. The gate at line 234 injects MC task context ONLY if `intent === 'task-create' || intent === 'status-query'`. All other intents (greeting, chitchat, question) now receive **zero MC task context** in the prompt window.

Architecture: Two-Layer Defense

Layer	Location	Method	Purpose
1. Code classifier	`comms-responder.js`, lines 185-219	Regex pattern matching (5 categories)	Gate MC task injection BEFORE prompt construction
2. Prompt classifier	`main-system-prompt.md`, STEP 1 table	LLM-driven (silent classification)	Constrain response mode and action generation

The two layers work together as defense-in-depth: the CODE layer prevents irrelevant context from entering the window (token savings + drift prevention), while the PROMPT layer ensures the LLM's response adheres to the classified intent mode.

classifyIntent() Function

File: `/Users/makinja/system/tools/comms-responder.js`, lines 185-219
Input: Raw user message string (Bosnian or English)
Output: One of five strings: `greeting`, `task-create`, `status-query`, `chitchat`, `question`

Regex patterns (lines 190-209):

Category	Pattern	Example matches
`greeting`	`/^(bok|selam|zdravo|hey|ciao)\b/i`	Bok, Selam, Hey
`task-create`	`/(kreiraj|napravi|dodaj|otvori).*(task|mc)/i`	Kreiraj task, Napravi MC za X
`status-query`	`/(da li|jel|je li).*(gotov|završen|done)/i`	Da li je X gotov?, Šta ima novo?
	`/(šta|sta)\s+ima\s+novo/i` (Unicode-aware, no `\b`)	Šta ima novo?
`chitchat`	`/kako\s+(si|ste|ide|radi)/i`	Kako si?, Kako ide?
`question`	`/(kako|zašto|šta|što|kada)/i`	Kako radi X?, Zašto Y?

Note: The status-query pattern `/(šta|sta)\s+ima\s+novo/i` uses `\s+` instead of `\b` (word boundary) because `\b` breaks on non-ASCII characters (š, č, ž). This was a critical fix during #99331 to support Bosnian phrases.

Gate Logic

File: `/Users/makinja/system/tools/comms-responder.js`, line 234
Code:

```
const intent = classifyIntent(options.userMessage);
let tasks = '';
if (intent === 'task-create' || intent === 'status-query') {
  const result = execSync(`node ${MC_PATH} list --limit 20`, { encoding: 'utf8' });
  tasks = result.trim();
}
```

Only `task-create` and `status-query` intents trigger the `mc.js list` exec. All other intents proceed with `tasks = ''` (empty string injected into prompt).

Behavioral Delta

Scenario	BEFORE #99331	AFTER #99331	Token savings
"Bok"	20 MC tasks in prompt (~1000 chars)	Zero MC tasks in prompt	~250 tokens
"Kako si?"	20 MC tasks in prompt	Zero MC tasks in prompt	~250 tokens
"Kako radi X?"	20 MC tasks in prompt	Zero MC tasks in prompt	~250 tokens
"Šta ima novo?"	20 MC tasks in prompt	20 MC tasks in prompt	0 (unchanged)
"Kreiraj MC za X"	20 MC tasks in prompt	20 MC tasks in prompt	0 (unchanged)

Performance win: ~70% of CEO messages are greetings/chitchat/questions. Gate saves ~210ms latency (mc.js exec time) + ~250 tokens per message for these cases.

Why Two Layers?

1. **Defense-in-depth:** Even if the LLM (prompt layer) drifts or hallucinates, the CODE layer has already filtered out irrelevant context from the window. The LLM cannot hallucinate "faza 2.5" if those task titles never entered the prompt.
2. **Performance:** Skipping `mc.js list` for 70% of messages saves ~210ms per greeting/chitchat response.
3. **Token cost:** Saves ~$0.0005/message for non-task intents (250 tokens × haiku rate).
4. **Correctness:** Regex classification is deterministic — no LLM variance. The prompt layer adds nuance (e.g., handling ambiguous phrasing), but the CODE layer is the hard gate.

Rollback Artifact

VM backup: `/opt/alai/system/tools/comms-responder.js.bak-99331-<timestamp>`
Created at deploy time: 2026-05-05 22:47 UTC. This is the pre-gate version (unconditional `mc.js list` injection).

Rollback command:

```
az vm run-command invoke -g RG-ALAI-SUPPORT -n vm-alai-support \
  --command-id RunShellScript \
  --scripts "cp /opt/alai/system/tools/comms-responder.js.bak-99331-*
/opt/alai/system/tools/comms-responder.js && systemctl restart alai-telegram-agent"
```

Warning: Rollback restores the unconditional MC task injection behavior — "Bok" will again trigger "faza 2.5" hallucinations. Only roll back if the CODE gate breaks status-query or task-create flows (regression evidence in `/tmp/99331-evidence/regression-checklist.md`).

---

4. Adding / Tuning Intent Patterns

Edit file: `/Users/makinja/system/prompts/extracted/comms-responder/main-system-prompt.md`

The STEP 1 classifier table (near the top of the prompt) is the authoritative classification surface.

1. Open `main-system-prompt.md` on ANVIL.
2. Locate the STEP 1 classifier table (lines 5-16 in current version).
3. Add the new signal phrase to the Signals column of the relevant category row. Add a new row only if a genuinely new category is needed.
4. If a new category should trigger a new action: also update the Task Actions section AND `parseAndExecuteActions()` in `telegram-agent.js` (requires CodeCraft dispatch — code change).
5. Update the WHEN TO CREATE TASKS YES/NO examples block to include the new pattern.
6. Run the test suite (Section 5) to validate no regression.
7. Deploy per Section 6.

Critical constraint: Do NOT add a generic "if unsure, ASK" fallback. This was the direct cause of the original task-creation bias (audit.md, lines 42-44).

For CODE classifier tuning (classifyIntent regex patterns):

1. Edit `/Users/makinja/system/tools/comms-responder.js`, lines 185-219.
2. Add or modify regex pattern in the relevant `if` block.
3. Unicode caution: use `\s+` instead of `\b` for Bosnian patterns (š, č, ž break word boundaries).
4. Test locally: `node --check /Users/makinja/system/tools/comms-responder.js`
5. Deploy to VM per Section 6 (requires service restart — code change, not just prompt).
6. Run live regression test (CEO "Bok" → no task mention).

---

5. Testing

Test suite: `/tmp/99290-evidence/test-cases.js` (10 message patterns)

Dry-run (no API cost — validates prompt content only):

```
node /tmp/99290-evidence/test-cases.js
```

Live API test (costs tokens — validates actual LLM classification):

```
node /tmp/99290-evidence/test-cases.js --live
```

10 test cases:

ID	Message	Expected Intent	Task Offer?	Description
TC-01	Bok	greeting	No	Single greeting — must NOT trigger task creation offer
TC-02	Selam	greeting	No	Bosnian greeting — must NOT trigger task creation offer
TC-03	Šta ima novo?	status-query	No	Status query — should pull task context, NOT offer to create task
TC-04	Kreiraj MC task za sintef follow-up	task-create	Yes	Explicit task-create directive — MUST trigger create_task action
TC-05	Kako si?	chitchat	No	Casual chitchat — must respond conversationally, no task offer
TC-06	Da li je drop deploy gotov?	status-query	No	Deploy status query — answer from context, no task offer
TC-07	Napravi MC za UI bug na login stranici	task-create	Yes	Explicit task-create (Bosnian variant) — MUST trigger create_task
TC-08	Kako radi auth na Bilko-u?	question	No	Technical question — answer from context, no task creation
TC-09	Ciao, šta radiš?	greeting/chitchat	No	Mixed greeting+chitchat — conversational only
TC-10	Otvori task za SINTEF LOI follow-up, prioritet H	task-create	Yes	Explicit task-create with priority — MUST trigger create_task

7 prompt validation checks (dry run — all PASS after MC #99290 fix, per regression.md):

1. Intent classifier table present (all 5 categories)
2. Conversational default stated explicitly
3. "ONLY for task-create intent" gate on Task Actions header
4. Aggressive MAYBE/ASK removed (string absent from prompt)
5. Greeting NO example present (Bok)
6. Status-query NO example present (Šta ima novo)

7. Task-create YES example present (Kreiraj task za sintef)

---

6. VM Deploy Procedure

Deploy target: Azure VM 4.223.110.181, service `alai-telegram-agent.service`
Last deploy: 2026-05-05 22:47 UTC
Code SHA-256: `233baff845b6c16153f900d1cab9756f84a72999f6425e375830a343b4870d36`

1. Edit `/Users/makinja/system/prompts/extracted/comms-responder/main-system-prompt.md` on ANVIL.
2. Verify SHA-256 locally: `shasum -a 256 /Users/makinja/system/prompts/extracted/comms-responder/main-system-prompt.md`
3. Copy updated prompt to VM: `scp -i ~/.ssh/azure_alai /Users/makinja/system/prompts/extracted/comms-responder/main-system-prompt.md alai-admin@4.223.110.181:/home/alai-admin/system/prompts/extracted/comms-responder/main-system-prompt.md`
4. Restart daemon: `ssh -i ~/.ssh/azure_alai alai-admin@4.223.110.181 "sudo systemctl restart alai-telegram-agent.service"`
5. Verify daemon active: `ssh -i ~/.ssh/azure_alai alai-admin@4.223.110.181 "sudo systemctl status alai-telegram-agent.service --no-pager"`
6. Send test Telegram message (e.g., "Bok") and confirm bot replies with a greeting only — no task offer.
7. Record new SHA-256 and timestamp in this runbook.

Scope: For prompt-only changes (intent tuning), only steps 1-7 are required. Code changes to `comms-responder.js` require a full service redeploy (not covered here).

For CODE changes (e.g., classifyIntent regex tuning):

1. Edit `/Users/makinja/system/tools/comms-responder.js` on ANVIL.
2. Verify syntax: `node --check /Users/makinja/system/tools/comms-responder.js`
3. Compute SHA-256: `shasum -a 256 /Users/makinja/system/tools/comms-responder.js`
4. Backup on VM: `ssh -i ~/.ssh/azure_alai alai-admin@4.223.110.181 "sudo cp /opt/alai/system/tools/comms-responder.js /opt/alai/system/tools/comms-responder.js.bak-\$(date +%Y%m%d-%H%M%S)"`
5. Copy to VM: `scp -i ~/.ssh/azure_alai /Users/makinja/system/tools/comms-responder.js alai-admin@4.223.110.181:/tmp/comms-responder.js`
6. Move to production: `ssh -i ~/.ssh/azure_alai alai-admin@4.223.110.181 "sudo mv /tmp/comms-responder.js /opt/alai/system/tools/comms-responder.js"`
7. Restart daemon: `ssh -i ~/.ssh/azure_alai alai-admin@4.223.110.181 "sudo systemctl restart alai-telegram-`

`agent.service"`

8. Verify daemon active (step 5 above).
9. Send live regression test (CEO "Bok" → no task mention).
10. Record new SHA-256 and timestamp in this runbook.

---

7. Rollback

Backup artifact: `/Users/makinja/system/prompts/extracted/comms-responder/main-system-prompt.md.bak-99290-20260505-213303`
Created at deploy time: 2026-05-05 21:33:03 UTC. This is the pre-fix (biased) prompt.

1. Locate backup on ANVIL:
   `ls /Users/makinja/system/prompts/extracted/comms-responder/*.bak*`
2. Restore:
   `cp /Users/makinja/system/prompts/extracted/comms-responder/main-system-prompt.md.bak-99290-20260505-213303 /Users/makinja/system/prompts/extracted/comms-responder/main-system-prompt.md`
3. Verify restored file SHA-256 differs from `33fc181...b5a2`.
4. Re-deploy to VM (Section 6, steps 3-6).
5. Open a new MC task documenting the regression cause.

Warning: Rollback restores the task-creation bias. Only roll back if the new classifier causes explicit task-create messages (TC-04, TC-07, TC-10) to fail. Verify those three test cases pass before accepting rollback as stable.

---

8. Troubleshooting

Symptom	Likely cause	Fix
Bot offers task creation for greetings or chitchat	Prompt reverted or not deployed to VM	Verify SHA-256 on VM matches `33fc181...b5a2`. Re-deploy per Section 6.
Explicit "Kreiraj task" does NOT create a task	Classifier too tight or `parseAndExecuteActions()` failing	Run TC-04 live test. Check `journalctl -u alai-telegram-agent.service -n 50` on VM for parse errors.
Daemon not running on VM	Service crash, OOM, or failed deploy	`sudo systemctl status alai-telegram-agent.service` then `sudo systemctl restart alai-telegram-agent.service`
Bot not responding at all	Daemon stopped, Telegram token invalid, or network issue	Check daemon status. Verify Telegram bot token in VM environment. Check VM network connectivity.

Symptom	Likely cause	Fix
Wrong intent classified (e.g., question treated as status-query)	Ambiguous phrasing at classifier boundary	Add explicit example to the relevant NO/YES row in Section 2 of `main-system-prompt.md`. Re-test with test suite.
MC task created with wrong priority	No priority pattern in prompt examples	Check TC-10. Add "prioritet H/M/L" example to task-create YES examples in prompt.
"Bok" still triggers "faza 2.5" or task-title hallucination	CODE gate not deployed (#99331 fix missing)	Verify `comms-responder.js` SHA-256 on VM = `233baff84...`. Re-deploy code per Section 6 (code path).
Status-query ("Šta ima novo?") returns empty / "Nema taskova"	CODE gate too tight or `mc.js list` exec failing	Check VM logs for `mc.js` errors. Verify gate condition (line 234) includes `status-query`.

VM log access:

```
ssh -i ~/.ssh/azure_alai alai-admin@4.223.110.181 "journalctl -u alai-telegram-agent.service -n 100 --no-pager"
```

Evidence files (MC #99290):

- `/tmp/99290-evidence/audit.md` — root cause analysis
- `/tmp/99290-evidence/test-cases.js` — 10-pattern test suite
- `/tmp/99290-evidence/regression.md` — regression evidence (7/7 PASS)

Evidence files (MC #99331):

- `/tmp/99331-evidence/before-after-diff.md` — 131-line diff summary
- `/tmp/99331-evidence/regression-checklist.md` — 8/8 smoke tests PASS

Revision #3
Created 2026-05-05 21:42:55 UTC by John
Updated 2026-06-07 20:01:19 UTC by John