LLMON explained: an LLM-native markup language that separates instructions from data in Granite 4.1

24 November 2026 · 8 min read · blogs.iictp.org/llmon-llm-native-markup-granite-prompt-injection-explained

Short answer: LLMON (LLM Object Notation, pronounced “lemon”) is an XML-like markup that wraps each part of a prompt in tagged spans delimited by reserved special tokens, so a model is told what each component means and which data an instruction applies to. IBM released two experimental preview models trained on it, Granite 4.1 3B and 8B, on 12 November 2026. IBM says indirect prompt injections in user data are mostly ignored, with attack success rates near zero on some benchmarks.

TL;DR The short version

- LLMON is IBM's XML-like markup that wraps each part of a prompt in tags made of reserved special tokens, telling the model what each part is and which data an instruction applies to.
- IBM says preview Granite 4.1 models trained on it mostly ignore indirect prompt injections hidden in user data.
- They are experimental previews; test that legitimate instructions inside data are not ignored too.

Read online, listen to it, or change the text size: https://blogs.iictp.org/llmon-llm-native-markup-granite-prompt-injection-explained

What is LLMON?

IBM’s post explains that a model’s context fills with many kinds of component: system prompt, earlier turns, retrieved documents, tool responses, output constraints and reasoning. Most of that structure is implicit, or encoded in formats such as JSON that were not designed for language models. IBM argues this is a reason smaller models are brittle, and that missing structure can be exploited by prompt injection.

How it works

- **Tags and special tokens.** Semantically coherent components get meaningful tags wrapped in reserved special-token delimiters, so structure and content use different parts of the vocabulary. Tags are stated at the start of each span and nested structure is stated locally.
- **Instruction binding.** Unique IDs let an instruction anywhere in the context point to the exact data it should act on, like a function call passing arguments.
- **Not a replacement for chat templates.** Chat templates say which role a part belongs to; LLMON says what each part means and how parts relate. The preview models ship their own chat template, and a part can be

marked exec: True or False, which controls whether it is bound into the exec block that says which instruction to run and what it operates on.

- **JSON conversion.** Converters turn existing JSON into LLMON and back without loss, and users can define their own tags and tune a LoRA on top.

Why not JSON?

IBM says JSON was designed for parsers, not for causal language models. The tokenizer does not respect the grammar, structure is carried by tokens such as braces and quotation marks that also occur in normal text, escaping rules are only statistical patterns to a model, and nesting depth is tracked by counting brackets, which transformers are said to be weak at over long spans.

Results reported by the source

- Indirect prompt injections hidden in user data are mostly ignored by LLMON models because of explicit instruction and data separation. On some benchmarks attack success rates drop to nearly zero, whereas regular instruction-tuned models are hijacked with high probability, according to IBM.
- General capability is said to be maintained relative to Granite 4.1 Instruct models.
- The release includes granite-4.1-3b-LLMON-preview and granite-4.1-8b-LLMON-preview and a GitHub toolkit with examples. The Mellea framework can LLMONize its model calls automatically.

Limitations and open questions

- They are labelled experimental preview models, and the benchmark names and exact numbers were not given in the passages we read, so the “near zero” claim should be checked on your own attack set.
- A model that ignores injected text may also ignore legitimate instructions placed inside data; test both directions.
- LLMON needs models trained to use it. It is not a prompt trick that works with any model.

How to try it

1. Load a preview model and build your usual messages list, naming each part and setting exec True or False.
2. Call apply_chat_template as usual; it renders the LLMON text.
3. Convert one JSON-based prompt with the converters and compare behaviour with the JSON version.
4. Run your own indirect prompt-injection tests, with instructions hidden in retrieved documents and tool outputs, before relying on the result.

What it means for developers and security teams

Structure in the prompt is a security boundary as well as a quality aid. LLMON does not remove the need for permission limits and monitoring. See what is prompt injection, prompt and context engineering and system-1 agents with local tool calling.

Frequently asked questions

- **What is LLMON?** An LLM-native markup language that uses reserved special tokens to mark the structure and meaning of prompt components.
- **Which models support it?** IBM's experimental granite-4.1-3b-LLMON-preview and granite-4.1-8b-LLMON-preview.
- **Does it stop prompt injection?** IBM reports that indirect injections are mostly ignored and near zero on some benchmarks; test it yourself.
- **Can I keep my JSON?** Yes; IBM provides lossless converters between JSON and LLMON.

Related reading

- What is prompt injection?
- Prompt engineering and context engineering
- Harness adapters: RL-trained LoRA for agent harnesses
- Local LLM Practitioner

Sources and currency

Sources were read on 24 November 2026. Products, rules and standards change, so check the linked originals before you rely on any detail.

> Related credential: Local LLM Practitioner. Explore the exam facts and syllabus.

Sources and further reading

1. IBM Granite team (Abraham Daniels, Basel Shbita, Chad DeLuca, Dan Gutfreund, Michael Hind, Shub A). LLMON: An LLM-Native Markup Language, now in Granite 4.1 3B and 8B. Hugging Face enterprise article, 12 November 2026. huggingface.co/blog/ibm-granite/llmon-an-llm-native-markup-language-now-in-granite.

Published by IICTP, an independent technical credential body, on 24 November 2026. The current version of this post is at https://blogs.iictp.org/llmon-llm-native-markup-granite-prompt-injection-explained. More guides: blogs.iictp.org.