AI Designer In Action

This page summarizes how AI can assist on creating solutions using the AI Designer tools.

How It Works

1. **Connect:** Claude Desktop connects to Designer via MCP
2. **Authorize:** First tool call triggers an MCP authorization prompt — approve it once
3. **Describe:** Tell AI what you need in plain English
4. **Watch:** Changes appear directly in Designer in real time
5. **Refine:** Continue the conversation to adjust

First-time build walkthrough: Download the AI Designer Guide

How It Works
Visual Feedback
How to Get the Best Results
What AI Can Configure
Not Just a Code Generator
AI Skills — Step-by-Step Guides
Learn FrameworX with the AI Tutor

Visual Feedback

When AI is connected, Designer shows:

- **"AI Designer" badge** — Orange label in the toolbar
- **Orange border** — Glowing border around the working area

You always know when AI is actively controlling the Designer.

How to Get the Best Results

The following practices reduce iteration and get you to a working solution faster.

Structure every prompt with Role, Context, and Instructions

This three-part structure is the most reliable way to improve first-attempt results. Use it at the start of every session and for any complex prompt.

- **Role:** Tell Claude what it is. Example: *"You are an automation engineer working in FrameworX by Tatsoft, specializing in SCADA systems."*
- **Context:** Describe your solution. Example: *"I have a bottling line solution called BottomLine with areas for filling, washing, and capping."*
- **Instructions:** State exactly what to do. Example: *"Create a device called Filler_PLC1 using Modbus TCP at 192.168.1.10 with the following tags..."*

Once you set the role in the first prompt of a session, you don't need to repeat it. Context and instructions carry over as the conversation continues.

Use FrameworX terminology

Claude understands general automation terms, but using FrameworX-specific names produces more accurate results without extra interpretation.

Instead of	Use
PLC, controller	**device**
Tag database	**UNS**
Screen, page	**display**
Data logging	**historian tag**
Folder, group	**area**

Know when to specify and when to let AI decide

You do not need to define everything upfront. Define what you know. For things you are unsure about, ask Claude to suggest options.

- **Define:** protocol, folder paths, data types, alarm setpoints, device addresses, IP addresses.
- **Let AI suggest:** color schemes, layout structure, icon choices, label formatting, object positioning, naming conventions.

For example, if you don't have a tag list yet:

"I'm building a water pumping station with two centrifugal pumps. I don't have the tag list yet. Suggest the most common tags for this equipment type, including data type and engineering units."

That prompt gets you a working starting point without requiring you to know everything upfront.

Work through the pillars in order

Build in this sequence: tags, devices, alarms, historian, displays, scripts. Each layer depends on the previous. Confirm each layer works before moving on.

Split backend and displays into separate sessions

For larger builds, complete all backend work (tags, devices, alarms, historian, scripts) in one session, then start a fresh Claude session for displays. This gives you a clean context window for display work and prevents running out of tokens mid-build. When starting the display session, briefly re-state your context so Claude knows the solution structure.

When building multiple displays, include this instruction in your prompt:

"After each display, stop and wait for my review before proceeding to the next."

This lets you catch issues display by display and keeps you in control of the pace.

When something doesn't work, tell Claude what's wrong

Do not restart the session or assume the tool is broken. Describe the problem in plain language. Claude can identify and fix most issues in the same session without losing the work already done.

If a session runs long and responses become degraded, start a new chat, re-state your context briefly, and continue from where you left off.

Current limitations

- **Display graphics:** AI-generated displays are structurally correct. The visual polish — element sizing, alignment, color consistency — typically needs manual adjustment in Designer.
- **Image placement:** Placing images on canvas displays is an area where AI does not produce reliable results. Place images manually and then ask Claude to work around them.

Example Prompts

New Solution Prompts

Specific prompts:

What You Say	What AI Does
"Create 10 temperature tags in the Boiler folder"	Creates Tag.Boiler/Temp1 through Temp10 with proper naming
"Connect to a Siemens S7-1500 at 192.168.1.100"	Searches protocols, creates Channel + Node with correct settings
"Build a dashboard showing all pump statuses"	Creates Dashboard display with cells for each pump
"Add high/low alarms for TankLevel"	Creates alarm items with proper conditions and groups
"Where is Tag.Flow1 used?"	Opens Find Results panel showing all usage locations
"Show me the Tags tab"	Navigates Designer to the UNS Tags view
"Delete the old TestDisplay"	Safely deletes the display after checking for references
"Rename Tag.OldName to Tag.NewName"	Renames with safe refactoring — all references update automatically
"Log in as admin"	Prompts for password, authenticates via designer_action('logon')

What AI Can Configure

Module	Objects
UNS	Tags, UserTypes (UDTs), TagProviders, Enumerations

Devices	Channels, Nodes, Points (all protocols)
Alarms	Groups, Items, Areas
Historian	Tags, Tables, Storage Locations
Displays	Canvas, Dashboards, Symbols
Scripts	Tasks, Classes, Expressions
Datasets	DBs, Queries, Tables, Files
Reports	Forms, WebData
Security	Users, Permissions, Policies

Not Just a Code Generator

Traditional Generators	AI Designer
Generate output, then done	Work side-by-side with AI continuously
Output goes to files	Changes appear directly in Designer
One-shot generation	Iterative conversation with full context
Limited to templates	AI understands your entire solution
No security awareness	Respects solution permissions and MCP category labels

AI Skills — Step-by-Step Guides

New for 10.1.5 — explicit skill discovery and run flow. When you ask Claude to discover or run a skill — phrases like *"show me FX skills"*, *"I want to do anomaly detection"*, or *"run the ML.NET regression skill"* — the AI now follows an explicit playbook: it calls `search_docs(query='', labels='skill', max_results=40)` to fetch the catalog, reads your recent conversation to infer what you're trying to do, and presents the 3–5 most relevant skills with escape hatches (type a number, type `all`, or describe your intent in natural language). Once you pick, the AI fetches the chosen skill's body, summarizes what it will do and what it needs from you, waits for your confirmation, then executes the MCP-call steps.

For specific requests like *"run the MQTT TagProvider skill"*, Claude searches by keyword, fetches the top match, confirms scope, and executes — no manual catalog lookup needed.

This flow works whether you've installed the optional FrameworX skill bundle (`frameworx.skill`) or you're using just the AI Designer MCP tools — the playbook is baked into both surfaces.

AI has access to **skills** — multi-module implementation playbooks that prevent common mistakes.

When you ask AI to configure something complex (e.g., Modbus TCP communication, historian logging), it automatically searches for relevant skills and follows step-by-step best practices.

Discover available skills:

> *"What skills are available?"*

AI uses: `search_docs('', labels='skill')` to list all published skills.

Learn FrameworX with the AI Tutor

If you want structured learning instead of ad-hoc help, the AI Tutor delivers a 45-lesson curriculum through Claude in your Designer MCP session. Three tiers — Essentials, Intermediate, Advanced — build from your first tag to enterprise multi-site architecture. Hands-on, adaptive, with progress tracking that survives version upgrades.

To start: ask Claude *"teach me FrameworX from the beginning"*, or *"start lesson E5"*, or *"I want to learn about alarms"*. The Tutor recognizes natural learning intent and routes to the right lesson.

See AI Integration for the full overview of FrameworX AI capabilities.

Get Started

Setup instructions: MCP and Claude Setup

Overview of all AI features: AI Integration

New Solution Prompts: New Solution Prompts

According to Anthropic's Claude

What Makes Our AI Unique

Best Practices for AI Designer

In this section...
