Enterprise Claw Agent

Autonomous AI Agent for Automation Anywhere

Agentic App Store Package Documentation

Version 2.15.0

Automation Anywhere

April 2026

1. Introduction

1.1 Overview

Enterprise Claw Agent is an Automation Anywhere package that brings autonomous AI agent capabilities directly into your agentic process automation workflows. It deploys an intelligent agent powered by Anthropic's Claude that can read documents, interact with web applications, write files, execute code, and complete complex multi-step business tasks — all orchestrated from a standard Automation Anywhere bot.

You give it a goal in plain English, and it autonomously figures out the steps to accomplish it — reading files, calling tools, browsing the web, filling out forms, and producing structured output.

1.2 Key Capabilities

Capability	Description
Document Intelligence	Reads text-based and scanned PDFs (with vision OCR), Word docs, and spreadsheets
Browser Automation	Navigates web applications via Playwright — fills forms, clicks buttons, reads pages using CSS selectors and accessibility tree
Desktop Control	Screenshots, mouse/keyboard control for desktop applications via computer use
Code Execution	Runs Python scripts and shell commands in a sandboxed workspace
Persistent Memory	Remembers facts across runs — learns preferences, caches domain knowledge
Custom Tool Library	Builds and reuses custom tools — creates them on the fly, loads them automatically on future runs
Structured Logging	Dual-format audit trail (human-readable + JSONL) with full token/cost accounting

1.3 Demo Use Cases

- Expense report automation — read receipt PDFs, navigate to expense system, submit reports
- Document generation — create formatted spreadsheets, reports, and summaries from raw data
- Research and analysis — search the web, compile findings, produce competitive analysis reports
- Data entry and form filling — read source documents and enter data into web or desktop applications
- File processing — batch rename, convert, organize, or transform files across directories
- System validation — check configurations, verify installations, generate compliance reports

2. Requirements & Prerequisites

2.1 Platform Support

Enterprise Claw Agent supports both Windows and macOS workstations. The Agentic App Store listing includes platform-specific packages:

Folder	Description
`EnterpriseClawAgent_Windows`	Package for Windows workstations (Bot Agent on Windows 10/11)
`EnterpriseClawAgent_macOS`	Package for macOS workstations (Bot Agent on macOS)
`DemoFiles`	Sample receipt PDFs for testing the included demo bot

2.2 Software Requirements

Automation Anywhere

- Automation Anywhere Control Room v.31 or later
- Bot Agent (Windows or macOS) v.20.11 or later
- User with Bot Runner or Bot Creator license

Workstation Dependencies

The following must be installed on the workstation where the bot will run. The Validate Workstation action will check for these and install missing Python packages automatically. (macOS instructions assume brew is installed)

Dependency	Windows Install	macOS Install
Python 3.10+	`winget install -e --id Python.Python.3.12`	`brew install python@3.12`
Claude Code CLI	`winget install -e --id Anthropic.ClaudeCode`	`npm install -g @anthropic-ai/claude-code`
Node.js (for CLI)	`winget install -e --id OpenJS.NodeJS.LTS`	`brew install node`

Anthropic API Key

An Anthropic API key is required for authentication. Obtain one from:

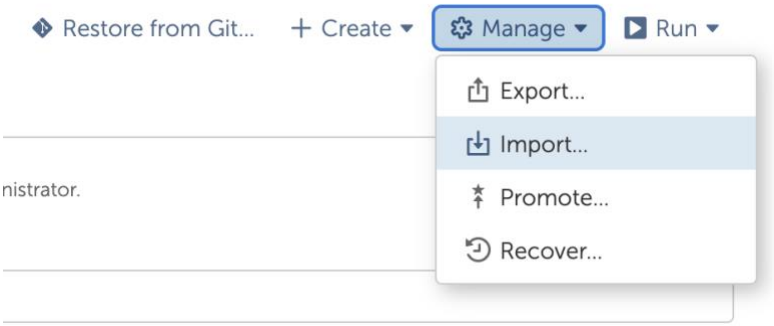
https://console.anthropic.com

The API key should be stored as a Credential in the Automation Anywhere Control Room credential vault for secure access.

3. Getting Started

3.1 Quick Start

1. Download the Enterprise Claw Agent package from the Agentic App Store
2. Install the ZIP file into your control room – the files will install to Agentic App Store\Enterprise Claw Agent\DemoFiles by clicking Manage > Import



3. Create a new TaskBot or execute on of the existing Demo automations.
4. Add a Run Agent action with your goal in plain English
5. Run the automation and review the results

3.2 Installation

Step 1: Upload the Package

1. Navigate to your Control Room → **Manage** → **Packages**
2. Click Add Package and upload the JAR file for your platform
3. The package will appear as **Enterprise Claw Agent** in your package list

Step 2: Store Your API Key

1. Navigate to Manage → Credentials
2. Create a new credential (e.g., “AnthropicAPIKey”)
3. Add an attribute and paste your Anthropic API key as the value
4. Assign the credential to the appropriate locker and users

Step 3: Leverage Demo Files (Optional)

The `DemoFiles` folder contains three sample receipt PDFs for testing:

- airportamys.pdf — Restaurant receipt
- teampizza.pdf — Pizza delivery receipt
- uber.pdf — Rideshare receipt

3.3 Actions Reference

Validate Workstation

Run this action once per machine before using Run Agent. It performs a complete environment check and setup:

- Detects and validates Python 3.10+
- Locates the Claude Code CLI
- Creates an isolated Python virtual environment
- Pre-installs 13 common libraries (PDF, spreadsheet, web, image processing, etc.)
- Extracts the agent runtime (bridge) to the workspace
- Writes bootstrap memory for the agent

Returns: A dictionary with a `message` key containing the full validation results.

Parameter	Type	Description
Workspace directory	Text (optional)	Path to workspace. Default: ~/aaclawagent

Run Agent

The primary action. Executes an autonomous AI agent loop with the specified goal.

Parameter	Type	Default	Description
Session name	Text	Default	Session identifier
Anthropic API Key	Credential	(required)	Your Anthropic API key from the credential vault
Goal / Task	Text Area	(required)	Plain English description of what to accomplish
Workspace directory	Text	~/aaclawagent	Directory for tools, memory, logs, and bridge runtime
Model	Select	Sonnet 4.6	Claude Opus 4.6, Sonnet 4.6, or Haiku 4.5
Max turns	Number	50	Maximum reasoning steps before the agent stops
Enable agent memory	Checkbox	On	Persistent cross-run memory (learns preferences, caches knowledge)

Feature Toggles

The Run Agent action includes optional feature toggles to enable additional capabilities:

Toggle	Default	Description
Web Access	Off	WebFetch and WebSearch — fetch URLs and search the web
Native File Editing	Off	Edit and MultiEdit — surgical in-place file edits

Jupyter Notebooks	Off	NotebookEdit — create and edit Jupyter notebooks
Task Lists	Off	TodoWrite and AskUserQuestion — task tracking and user prompts
Sub-agents	Off	Agent — spawns nested agents with separate context (uses more tokens)
Computer Use	Off	Screenshot, click, type, scroll — desktop application control
Browser Automation	Off	Playwright — navigate, click, fill forms via CSS selectors
Verbose Logging	Off	Thinking, tool timing, and progress — for demos and debugging

Return Values

The Run Agent action returns a dictionary with the following keys:

Key	Description
`status`	"success" or "failed"
`output`	The agent's final response text
`exit_code`	0 (success), 1 (warning), 2 (fatal error)
`run_id`	Unique identifier for this run
`log_path`	Path to the detailed audit log (.log + .jsonl)
`memory_path`	Path to the memory directory
`tools_created`	Names of any new custom tools created during this run

3.4 Built-in Tools

The agent has 11 built-in tools available on every run, regardless of feature toggles:

Tool	Purpose
`read_pdf`	Extract text from any PDF — handles scanned documents with vision OCR automatically
`list_directory`	List files and subdirectories
`read_file`	Read the contents of a text file
`write_file`	Write content to a text file
`run_shell`	Execute shell commands
`run_python`	Execute Python code
`remember`	Store facts in persistent cross-run memory
`recall`	Retrieve facts from persistent memory
`list_tools`	List all available custom tools in the tool library
`create_tool`	Create a new reusable custom tool (persisted to the tool library)
`use_tool`	Execute a custom tool from the tool library

3.5 Workspace Structure

Each agent run operates within a persistent workspace (~/aaclawagent by default):

Directory	Purpose
`tools/`	Persistent tool library — auto-loaded every run
`memory/`	Markdown knowledge base — persists across runs
`logs/`	Per-run audit logs (.log + .jsonl)
`bridge/`	Agent runtime (managed automatically)
`venv/`	Isolated Python environment

3.6 Demo Bot Walkthrough

The package includes a sample bot that demonstrates end-to-end expense report processing. Here is how the bot flow works:

Bot Flow Steps

4. **Assign workspace path** — Sets a string variable with the workspace directory path (e.g., ~/aaclawagent or C:\aaclawagent)
5. **Check if folder exists** — Uses an If condition to check whether the workspace/receipts folder already exists
6. **Validate Workstation** — Runs the Enterprise Claw Agent Validate Workstation action to check all dependencies and set up the environment
7. **Copy receipt files** — Copies the DemoFiles (3 sample receipt PDFs) from the Control Room file repository to the local workspace/receipts folder
8. **Run Agent** — Executes the Enterprise Claw Agent with a goal like: "Read the receipt PDFs in the receipts folder, then go to our expense system and submit them as a new report."
9. **Loop through results** — Iterates over the returned dictionary keys to log or process each result value
10. **Clean up** — Optionally deletes the temporary receipts folder

Demo Files

The DemoFiles directory contains three sample receipt PDFs that the demo bot uses:

File	Description
`airportamys.pdf`	Airport Amy's restaurant receipt — sample food/beverage expense
`teampizza.pdf`	Team pizza delivery receipt — sample team meal expense
`uber.pdf`	Uber rideshare receipt — sample transportation expense

These files can be uploaded to your Control Room file repository and referenced in the bot's Copy Files step.

3.7 Logging & Observability

Every agent run produces two log files in the workspace's logs/ directory:

- **`.log`** — Human-readable audit trail with thinking, tool calls, reasoning, and timing
- **`.jsonl`** — Structured events for tooling, SIEM, or automated analysis

The run summary at the end of each log includes: status, total turns, runtime, API time, total cost, and detailed token breakdown (input, cache read, cache create, output).

4. Security

- API credentials are stripped from subprocess environments — child processes spawned by run_shell and run_python cannot access the API key
- All file operations are scoped to the workspace and authorized directories
- Credentials are never written to logs or memory files
- The Automation Anywhere operator controls what the agent can do via the goal and feature toggles they configure

5. Support & FAQs

5.1 Frequently Asked Questions

Q: What models are supported?

A: Claude Opus 4.6 (most capable), Claude Sonnet 4.6 (recommended balance of speed and capability), and Claude Haiku 4.5 (fastest, lowest cost). Sonnet 4.6 is the default.

Q: How much does it cost per run?

A: Costs vary by model, task complexity, and number of turns. A typical 10-turn Sonnet run costs approximately $0.10–$0.50. The agent's prompt caching significantly reduces costs for multi-turn conversations. Detailed cost accounting is included in every run's log.

Q: What happens if the agent exceeds the max turns?

A: The agent stops gracefully and returns whatever progress it has made. You can increase max turns for complex tasks.

Q: Can the agent remember things between runs?

A: Yes. When memory is enabled (default), the agent persists knowledge to Markdown files in the memory/ directory. It automatically loads this knowledge at the start of each run.

Q: Does the agent work offline?

A: No. The agent requires internet access to communicate with the Anthropic API. The workstation must be able to reach api.anthropic.com.

Q: Can I use this on both Windows and macOS?

A: Yes. Upload the platform-specific package (EnterpriseClawAgent_Windows or EnterpriseClawAgent_macOS) that matches your Bot Agent's operating system.

Q: What if Validate Workstation fails?

A: Review the returned message dictionary for specific error details. Common issues: Python not installed, Claude Code CLI not found, or insufficient permissions. The validation output includes actionable instructions for each check.

5.2 Troubleshooting

Issue	Solution
"Python not found"	Install Python 3.10+ and ensure it's on the system PATH. Run Validate Workstation again.

"Claude Code CLI not found"	Install the Claude Code CLI (see Requirements section). On macOS, ensure /usr/local/bin or ~/.local/bin is on PATH.
"Bridge script not found"	Run Validate Workstation first — it extracts the bridge runtime to the workspace.
API key errors	Verify your API key at console.anthropic.com. Ensure the credential is correctly configured in the Control Room vault.
Agent times out	Increase max turns. The timeout scales automatically: 30 seconds per turn, minimum 300s, maximum 3600s.
Computer Use permissions (macOS)	Go to System Settings → Privacy & Security → Accessibility and Screen Recording. Add your terminal or Bot Agent application.

5.3 Support

For issues, questions, or feedback:

- Check the run logs in the workspace's logs/ directory for detailed error information
- Visit the GitHub repository: https://github.com/AutomationAnywhere/Enterprise-Claw-Agent
- Contact Automation Anywhere support for Control Room or Bot Agent issues

6. Appendix

6.1 Record of Changes

Version	Date	Changes
2.15.0	April 2026	Initial public release on the Agentic App Store. Includes Run Agent and Validate Workstation actions, 11 built-in tools, computer use, browser automation, persistent memory and tool library.

6.2 License

Enterprise Claw Agent is released under the Apache License 2.0. You are free to use, modify, and distribute this package.

The Claude Agent SDK (used internally) is licensed under the MIT License. The Claude Code CLI is a proprietary Anthropic product subject to Anthropic's Commercial Terms of Service.

6.3 References

Enterprise Claw Agent: https://github.com/AutomationAnywhere/Enterprise-Claw-Agent
Automation Anywhere Agentic App Store: https://botstore.automationanywhere.com