Kill Switch and Containment for AI Agents

Microsoft Foundry and Agent 365: what is possible today, what it looks like, and how to configure it

Type: Technical fact file and workshop brief **Scope:** Microsoft Foundry Agent Service, Entra Agent ID, Sentinel **Verified:** November 2026 **Version:** 1.0

This is a working reference for a workshop on enabling Agent 365 safely. It sets out what a kill switch for an AI agent actually is on the Microsoft platform, the exact mechanisms, the caveats that bite, how it is graded and triggered, who holds the authority, the network and telemetry that make it work, and what is generally available today versus still in preview. Feature and status details were verified against Microsoft documentation in November 2026; preview functionality changes, so reconfirm before a production change.

1. What a kill switch actually is

There is no single "kill" button. A kill switch for an agent is **three coordinated actions**, and a credible design uses all three, because each on its own leaves a gap:

1. **Stop the runtime** (Foundry Agent Service): end the agent's running process and active sessions so it stops doing work immediately.
2. **Disable the identity** (Entra Agent ID): block the agent from authenticating and being issued tokens, so it cannot act even if a process restarts.
3. **Cut the network path** (VNet / Azure Firewall): remove the agent's ability to reach data and egress, as containment and as a backstop while the first two propagate.

> **THE DESIGN POINT**
> Runtime-only leaves the identity live (it can be re-invoked). Identity-only leaves a running process holding a valid token until that token expires. Both together, plus a network cut, is what actually stops an agent. The order in an incident is usually: disable identity, stop runtime, then confirm network containment.

2. The three mechanisms, in detail

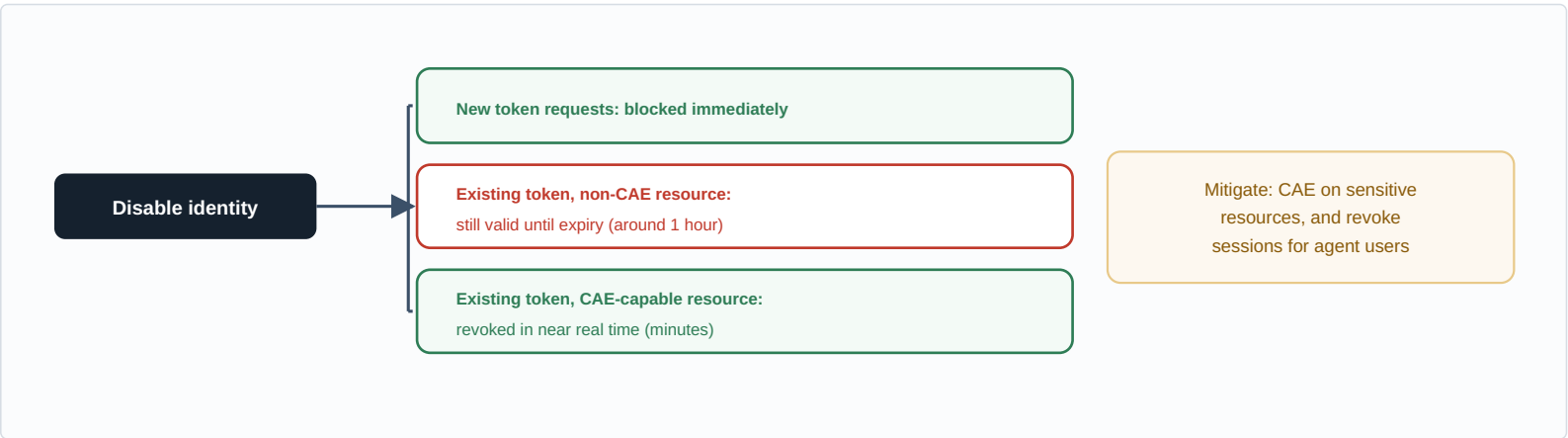
2.1 Disable the identity, Entra Agent ID

Agent identities are a distinct identity type in Entra ID. There are three scopes of disable, from surgical to blunt:

Scope	Action	Effect
Single agent	Entra ID > Agents > Agent identities > select > **Disable**. API: Graph `PATCH` the agent identity with `accountEnabled:false` (scope `AgentIdentity.ReadWrite.All`).	Blocks that agent's sign-ins and token issuance across Entra and connected apps.
Blueprint	Entra ID > Agents > Agent blueprints > select > **Disable** (optionally "Disable in all tenants"). The blueprint is a service principal; disabling it blocks new tokens for every agent identity and agent user built from it.	Kills all instances of that agent type at once.
Tenant-wide	Conditional Access policy to block agent identity authentication (needs Entra ID P1). Block creation by removing Agent ID Administrator and Agent ID Developer role assignments and the product-level creation controls.	Stops all agents, and new agent creation. Blunt, high blast radius.

An **agent user** (the separate user account some agents hold for systems that require a user) must be disabled in its own right. Conditional Access is currently weak at blocking agent users, so the reliable action is to disable the agent user account directly. **Disable is reversible** (re-enabling restores access and tokens); **delete is permanent**. For containment, disable and investigate, then re-enable or retire.

> **CAVEAT THAT BITES: LIVE TOKENS**
> Disabling blocks **new** tokens immediately, but an access token already issued stays valid for its lifetime (around one hour) **unless the resource supports Continuous Access Evaluation (CAE)**, which revokes in near real time. So a "disabled" agent can keep acting for up to an hour against non-CAE resources. Mitigations: ensure target resources are CAE-capable, and revoke active sessions (for agent users, `Revoke-MgUserSignInSession`). CAE becomes mandatory for new tenants from 1 October 2026. Treat the token-propagation gap as a known residual risk in any runbook.



Disabling blocks new tokens at once, but an already-issued token survives up to its lifetime unless the resource supports Continuous Access Evaluation.

2.2 Stop the runtime, Foundry Agent Service

Foundry Agent Service (GA) runs agents, including hosted containerised agents. The runtime kill is a delete with force:

- **Delete the agent and its active sessions:** `project.agents.delete(agent_name="...", force=True)`. `force=True` also tears down active sessions, not just the registration.
- **Delete a specific version:** pass `agent_version` (agent updates are published as immutable versions).
- **Delete a conversation / session:** `conversations.delete(conversation_id=...)` to end a specific live interaction.

Deleting the Foundry agent stops it serving, but it does **not** disable the Entra identity. That is why step 2.1 is separate and necessary.

2.3 Cut the network path

Covered in section 7. In an incident it is the fast backstop: remove the agent subnet's egress and its private-endpoint access to data while identity and runtime actions propagate.

3. Licence to operate, dev to pre-production to production

Do not enable agents in production without having proven the kill switch earlier in the pipeline. A practical progression, with what to demonstrate at each gate before promotion:

Stage	Agent scope	Kill switch to prove before promotion
Development	Low-privilege, synthetic or public data only, basic setup acceptable	Manual kill works end to end: disable the agent identity, delete the runtime with `force=True`, confirm the agent stops. Telemetry reaches Application Insights. Owner assigned.
Pre-production	Representative data, Standard Setup with private networking, production-like identity	Automated kill works: a Sentinel analytic fires a playbook that disables the identity and deletes the runtime, with an approval gate. Measure propagation time (including the token gap). Egress cut tested. Runbook rehearsed.
Production	Live data and actions, graded by privilege	Kill switch exercised on a schedule (game day), evidence retained. Named on-call authority. CAE confirmed on sensitive resources. Monitoring and alerting live.

> **GATE RULE**
>
> An agent does not promote to the next stage until its kill switch has been exercised and evidenced at the current one. An untested kill switch is an assumption, not a control. "We have a playbook" is not evidence; "we ran it last Tuesday and it stopped the agent in N minutes" is.

4. Grading the kill switch

Not every agent needs the same kill switch. Grade it to the agent's privilege and the damage it could do, so effort is spent where it matters.

Tier	Agent example	Kill switch	Trigger & approval
Low	Reads public web, reports back	Manual disable + delete, best-effort speed	Analyst-initiated, no automation required
Medium	Reads internal data, drafts	Semi-automated playbook, identity + runtime	SOC-triggered with approval; minutes
High	Writes to systems, moves money, changes config	Fully automated, identity + runtime + egress cut, CAE-enforced resources	Auto-contain on high-confidence trigger, approval to restore; seconds to low minutes

The grade sets three things: how fast the switch must act, how much is automated versus manual, and whether containment happens before or after human approval. High-privilege agents justify auto-containment with approval-to-restore; low-privilege agents do not.

5. What triggers a kill switch

Triggers come from three places. Each should map to a defined response (alert, contain-with-approval, or auto-contain) based on the agent's grade.

Source	Example triggers
Identity signals (Entra ID Protection for agents)	Unfamiliar resource access, sign-in spikes, repeated failed access, anomalous location or pattern. Entra ID Protection detects offline risk types for agent identities.
Behaviour and intent (telemetry, Sentinel)	Tool calls outside the agent's remit, access to data it should not touch, drift toward an unsafe action or command sequence, abnormal volume, spend or loop anomalies, prompt-injection indicators in agent behaviour.
Operational and manual	A named owner or SOC analyst pulls the switch; a change or incident declares the agent in scope; cost circuit-breaker; a failed integrity or policy check.

A useful rule: the trigger should express **intent drift** (the agent is heading somewhere it should not), not only that an action happened. That is the difference between a noisy switch and a trusted one.

6. Who has the authority to kill

Define this before go-live, at two levels, because they use different roles and have different blast radius:

Level	Who, and what they need
Agent level (one agent or blueprint)	The agent's **named owner** and the **SOC on-call**. To disable an agent identity or blueprint they need the **Agent ID Administrator** role (or delegated equivalent) in Entra; to delete the runtime they need the Foundry project role that permits agent management. Least-privilege: grant these to specific people, not broadly.
Platform level (tenant-wide)	A senior identity or security authority (for example a Global or Security Administrator with change authority). Tenant-wide actions, a Conditional Access block on all agent authentication, or removing Agent ID role assignments to stop creation, are high blast radius and should require named authority and a logged decision.

> **DECISION FOR THE WORKSHOP**
>
> Name the people, not the roles, for each tier, and agree the approval chain for auto-contain and for restore. Agree who can pull the platform-level switch, which stops every agent at once, and under what declared condition.

7. Network: ingress and egress

Foundry Agent Service offers two postures. Only the second gives you private egress and a network cut as a containment lever.

Posture	What it gives you
Basic setup	Public internet access. Inbound can be restricted with a private endpoint to the Foundry resource, but agent egress is not privately isolated. Not suitable for data-classified workloads.
Standard Setup with private networking	Bring-your-own VNet with VNet injection. No public egress, agent traffic stays off the public internet. Private endpoints and private DNS for Foundry, Azure Storage, Azure AI Search and Azure Cosmos DB (bring your own; these private endpoints are not auto-created). Private networking extends to tool connectivity: MCP servers, AI Search and Fabric data agents over private paths.

Ingress: private endpoint to the Foundry data-plane endpoint; clients reach the agent over Private Link. **Egress:** controlled with Azure Firewall on the agent subnet. The firewall allow-list must include the identity and login FQDNs the agent needs (for example `control-{region}.identity.azure.net`, `{region}.login.microsoft.com`, `login.microsoftonline.com`) or the `AzureActiveDirectory` service tag. **MCP and tool auth** over private networking supports key, Entra Agent Identity, Managed Identity and OAuth identity passthrough.

> **TWO NETWORK CAVEATS**
>
> **(1)** Some tool invocations are made by Agent Service through an internal mechanism that **bypasses the egress subnet entirely**, so an egress firewall does not see or control every outbound action. Do not assume the firewall is a complete egress control. **(2)** The Azure Container Registry holding hosted-agent images generally must remain reachable; a fully private ACR with public access disabled is not universally supported and depends on when the project was created. Confirm both against your setup.

8. Observability and monitoring to Sentinel

You cannot trigger or justify a kill switch on behaviour you cannot see. Foundry uses OpenTelemetry and Azure Monitor Application Insights as its telemetry backbone.

- **Server-side telemetry (automatic):** for hosted agents, Foundry Agent Service emits agent-invocation telemetry automatically, no code changes, via the hosting libraries that integrate the Microsoft OpenTelemetry distro.
- **What it captures:** agent reasoning, model calls, tool invocations, multi-agent interactions, timestamps, latency, token usage and errors. Spans carry a `gen_ai.agent.id` attribute so activity ties back to a specific agent.
- **Where it lands:** Application Insights, which is backed by a Log Analytics workspace, and is viewable in the Foundry Observability > Traces view.

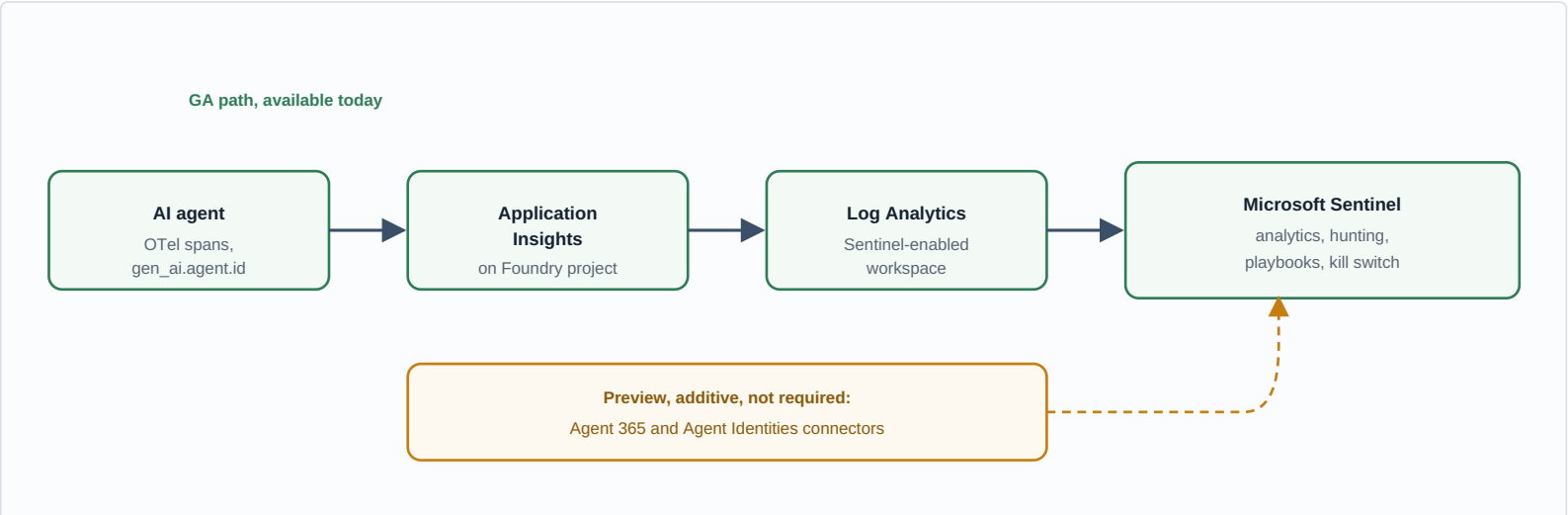
Getting it into Sentinel: connect the Application Insights resource to a Log Analytics workspace that is **Sentinel-enabled** (or ingest into one). The agent traces then land in standard Log Analytics tables (for example `AppTraces`, `AppDependencies`, `AppRequests`) where Sentinel analytics rules, hunting, workbooks and playbooks can act on them. Correlate on `gen_ai.agent.id`, model, tool call, session and the user or workload identity on whose behalf the agent acted.

9. The OpenTelemetry route, because the connectors are preview

> **THE KEY ANSWER**
>
> The native Agent 365 and Agent Identities connectors into Sentinel are still **public preview**. The OpenTelemetry path is **generally available** and does not depend on them. So you do not have to wait: instrument agents with the Microsoft OpenTelemetry distro, export to Application Insights connected to the Foundry project, and point that at the Sentinel-enabled Log Analytics workspace. That gives you agent observability and a kill-switch trigger surface today, on GA components.

How to use it:

- **Hosted agents:** server-side invocation telemetry is automatic. Just connect Application Insights to the Foundry project (set it up in Foundry tracing), then connect that Application Insights to the Sentinel workspace.
- **Custom or framework agents (client-side tracing, preview):** add the Microsoft OpenTelemetry distro (`azure-monitor-opentelemetry`), set `APPLICATIONINSIGHTS_CONNECTION_STRING` to the project's Application Insights, and spans for model and tool calls flow automatically. Microsoft Agent Framework and Semantic Kernel have built-in tracing; LangChain and LangGraph via `langchain-azure-ai`; OpenAI Agents SDK via its instrumentor.
- **What the preview connectors add later:** richer agent inventory and identity context (agent and blueprint inventory, ownership, unified agent observability). Useful, but additive. The OTel route carries the behavioural signal the kill switch needs now. Record the preview connectors as a planned enhancement with a review date tied to their general availability.



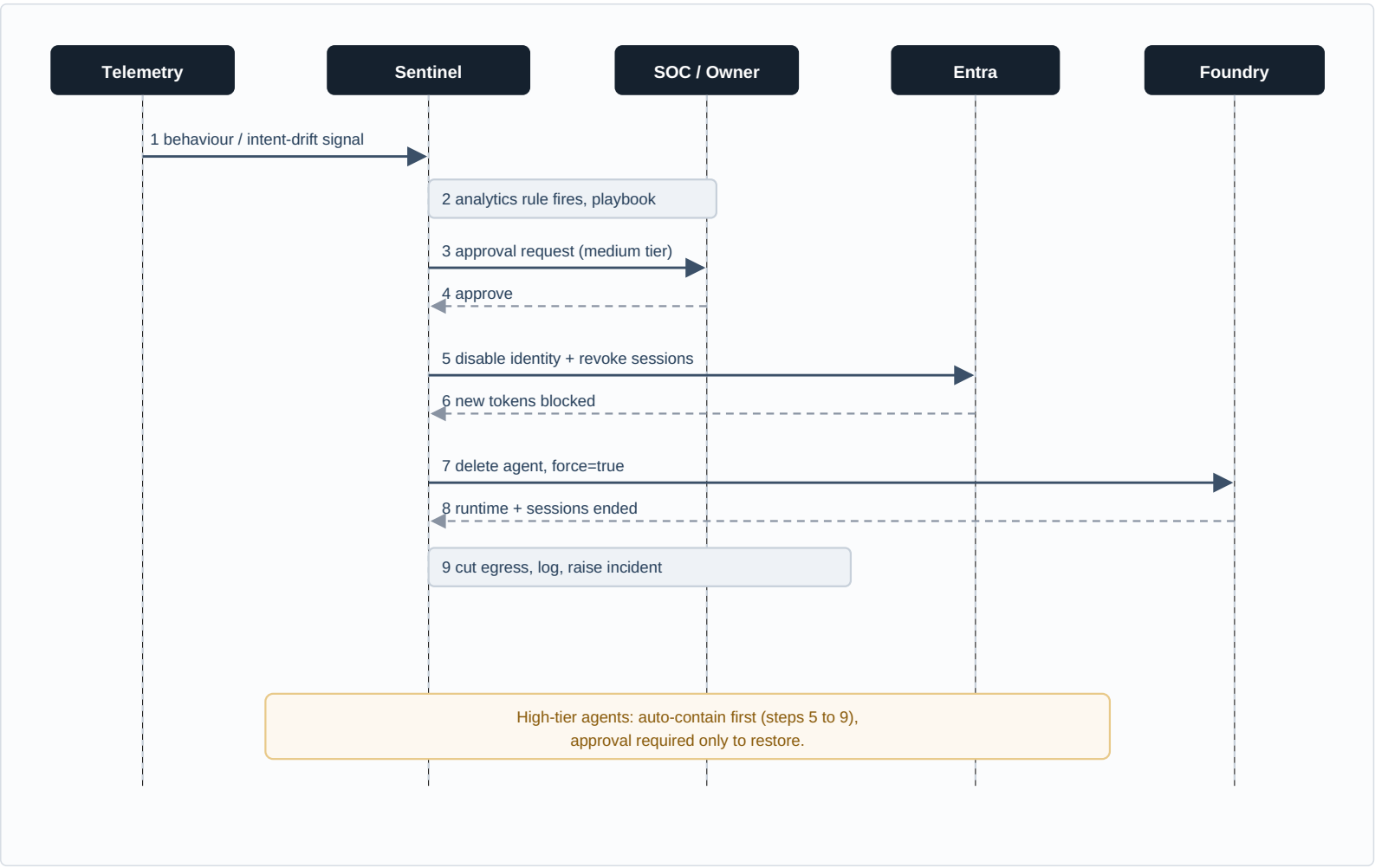
The OpenTelemetry route into Sentinel is generally available and connector-independent. The preview connectors add inventory and identity context on top.

10. SOC playbooks and runbooks

The kill switch is operationalised as a Sentinel automation (the automated path) plus a documented manual runbook (the fallback and the high-authority path).

Automated path (Sentinel SOAR)

1. **Detect:** a Sentinel analytics rule fires on an identity, behaviour or intent-drift signal (section 5).
2. **Orchestrate:** an automation rule invokes a playbook (Logic App).
3. **Approve (grade-dependent):** for medium-tier agents, an approval step (for example a Teams adaptive card to the owner or SOC lead). For high-tier agents, auto-contain first, approve to restore.
4. **Contain:** the playbook calls Microsoft Graph to set `accountEnabled:false` on the agent identity (and revokes sessions for any agent user), calls Foundry to delete the agent runtime with `force=True`, and optionally triggers the network egress cut.
5. **Record:** log the containment to an immutable store and raise the incident. Restore requires a separate approval.



The automated kill switch: a Sentinel analytic fires a playbook that disables the identity, deletes the runtime and cuts egress, with an approval gate graded by tier.

Manual runbook (fallback and platform-level)

1. Preserve evidence first: capture the agent identity, blueprint, object IDs, current status, owner, and the triggering telemetry.
2. Disable the agent identity (or blueprint, if the whole type is implicated). Disable any agent user.
3. Delete the Foundry runtime with `force=True`; end live conversations.
4. Confirm containment: new tokens blocked, running sessions gone, egress cut, and verify behaviour has stopped in telemetry (mind the token-propagation gap).
5. Decide: investigate, then re-enable, re-scope, or retire. Retire by deleting the identity only once no workload depends on it.

11. What is possible today versus roadmap

Capability	Status	Note
Entra Agent ID as a distinct identity type	GA	Individual, blueprint and tenant-wide disable available.
Disable agent identity / blueprint (portal + Graph)	GA	The identity-layer kill. Reversible; delete is permanent.
Conditional Access for agent identities	GA	Needs Entra ID P1. Weak at blocking agent users today.
Continuous Access Evaluation (near-real-time revocation)	GA	Resource must be CAE-capable. Mandatory for new tenants from 1 Oct 2026.
Entra ID Protection risk detection for agents	GA	Offline risk types; a trigger source.
Foundry Agent Service runtime, delete with `force=True`	GA	The runtime-layer kill, ends active sessions.
Standard Setup with private networking (no public egress)	GA	Required for private egress and the network-cut lever.
OpenTelemetry server-side telemetry (hosted agents)	GA	Automatic; to Application Insights.
OpenTelemetry to Application Insights to Sentinel workspace	GA	The route to use now; connector-independent.
Sentinel playbooks (Logic Apps) and automation rules	GA	The SOAR engine for the automated kill switch.
Foundry client-side tracing (framework instrumentation)	Preview	Works now; preview terms apply.
Agent 365 connector to Sentinel (UnifiedAgentObservability)	Preview	Richer agent activity; additive to the OTel route.
Agent Identities connector to Sentinel (inventory, blueprints)	Preview	Agent and blueprint inventory and ownership context.
Defender XDR agent threat detection via Agent 365	Preview	Another trigger source once GA.

The headline for the room: every component of a working kill switch, identity disable, runtime delete, private networking, OpenTelemetry to Sentinel, and SOAR playbooks, is **generally available today**. The preview items (the native connectors, Defender agent detection, client-side tracing) make it richer and easier, but you can build and operate the kill switch now without them.

12. Decisions to close in the workshop

- **Posture: are we mandating Standard Setup with private networking for any agent above low tier? (Needed for the egress cut.)**
- **Grading: what is the privilege-to-kill-switch mapping, and who classifies each agent?**
- **Authority: named people for agent-level and platform-level kill, and the approval chain for auto-contain and restore.**
- **Token gap: which sensitive resources must be CAE-capable before a high-tier agent is allowed to touch them?**
- **Telemetry: confirm the OpenTelemetry to Sentinel route now, and treat the preview connectors as a later enhancement.**
- **Egress caveat: how do we compensate for tool calls that bypass the egress subnet? (Identity and runtime controls carry more weight because of this.)**
- **Evidence: what is the game-day cadence and the evidence standard for the licence to operate at each stage?**

Technical fact file, version 1.0, November 2026. Mechanisms, roles, API calls and GA/preview status verified against Microsoft documentation at the date shown. Preview functionality and feature names change; reconfirm current status, role names and API versions before a production change. API and PowerShell references are illustrative of the documented approach, not a substitute for the current Microsoft reference.