SaaCIP 0.2: the Hosted SaaC profile (draft)

Imtiaz Hassan, The Software Lens

The living version of this document is at https://thesoftwarelens.com/saacip/hosted/. This PDF is a dated snapshot, for citing; it is never edited after publication.

SaaCIP 0.2: the Hosted SaaC profile

Profile name	`hosted`
Protocol	`saacip/0.2`, defined in SaaCIP: the SaaC Interoperability Protocol (the core)
Status	Version 0.2, a draft for comment. Not a standard. The first version of this profile; it answers open point 1 of SaaCIP 0.1
Date	6 November 2026
Author	Imtiaz Hassan
Binding	HTTPS with JSON: the binding of the Resident profile, section 6, used unchanged
Companion document	SaaCIP 0.2: the Resident SaaC profile, whose connector manifest, declared action form and HTTP binding this profile reuses
Machine-readable parts	The same schemas as the Resident profile, under /saacip/0.2/; a Hosted example terms document beside them
Dated snapshot	SaaCIP-0.2-hosted-profile-draft.pdf, for citing; this page is the living version
Licence	Specification text: CC BY 4.0. Schemas, examples and validator: Apache License 2.0, as for the core

Abstract

A **Hosted SaaC** is a capability that runs at the provider's and is hired by an organisation whose hub stays in its own environment. It is the shape most vendors will offer first, because it is the shape they already run: the service lives with the provider, and the hirer's data goes to it. What SaaCIP adds is the hire around that shape: terms that say what goes, what comes back and which model sees it; a hub the hirer runs, which the capability calls and which never calls in; receipts for every step; a person's judgement on the work; a statement at the end, and the hub's attestation of it.

This profile is honest about one thing the Resident profile does not have to be: in the Hosted shape, residency is not a property of the architecture, it is a property of the terms, and some of the evidence is the capability's own account. Section 3 says exactly which evidence the hirer can check from its own side and which it cannot, so that a hirer chooses this profile knowing the difference.

This document only makes sense with the core and the Resident profile beside it. Everything about the hire, its documents, its lifecycles, its autonomy levels, the connector manifest, the form of a declared action and the HTTP binding is defined there and is not repeated here.

Status of this document

As for the core: ongoing research, published for comment, not a standard, not legal advice.

Licence and patents

As for the core: the text is licensed under CC BY 4.0; the schemas, examples and validator under the Apache License 2.0. The author holds no patents on SaaCIP and has not applied for any.

Contents

- 1. What a Hosted SaaC is

- 2. Conventions
- 3. The evidence split: what the hirer can check
- 4. The four rules of hosting
- 5. Connectors, records and declared actions
- 6. Credentials
- 7. The HTTP binding
- 8. Operating rules
- 9. Conformance
- Appendix A. Where attestation of the provider's environment would plug in
- Appendix B. Change log

---

1. What a Hosted SaaC is

A hirer with data that may leave its environment, or a job that does not need to act inside it, has no reason to run the capability itself. The provider runs it, as a service, and the hirer sends it what the job needs: a nightly export, a stream of events, a folder of documents, read through a connector the provider operates with credentials the hirer issued. The capability does its work at the provider's, reports to the hirer's hub, asks the hub before it acts, and leaves receipts as any capability does.

The hub stays at the hirer's. That is the one fixed point of every SaaCIP profile: the place where a hire is created, where people decide, and where the record of the hire is kept, belongs to the hirer. A provider that also offers to run the hub for its customers is offering something SaaCIP does not cover, because the hirer would then be judging the capability on the capability's own system of record.

Everything the core defines still applies: the hire has the same kinds and lifecycle, the same terms, receipts, outcomes, statements and attestations, the same autonomy levels and the same declared actions file. The direction of every call is the same as in the Resident profile, outbound from the capability, so the hirer opens nothing. What differs is where the data is read and where the capability's state lives, and therefore what the hirer can verify. Section 3 is the heart of this profile.

Example. A provider offers `example-timesheet-check`, a capability that keeps a hirer's timesheets complete and correct. The hirer hires it on a 14-day trial. Each night the hirer's time-logging system pushes the week's entries to the capability through a connector the hirer authorised with a short-lived, read-only token. The capability finds eleven people with no entries for Thursday and three entries that look wrong, reports them to the hirer's help desk under refs of its own, and asks to send reminders, a declared action; a person approves, the reminders go out through the hirer's own mail system with a separate credential, and the receipts say which credential did what. The person judges the findings, the trial ends with a statement, and the hub attests it.

2. Conventions

The requirement words MUST, SHOULD, MAY and their relatives have the meanings given in the core, section 2.1. Terms not defined here have the meanings given in the core, section 2.2, and in the Resident profile. "The core" means SaaCIP 0.2. In the binding the word *agent* always means the capability.

3. The evidence split: what the hirer can check

In the Resident profile the hirer owns the network, so every byte that leaves is visible to it. In the Hosted profile the hirer's data crosses to the provider, and from there the hirer sees nothing directly. This profile

does not pretend otherwise. It divides the evidence of a hire into what the hirer can check from its own side and what is the capability's own account, and it requires both to be labelled as such.

Evidence	Who can check it, and how
What the hirer sent to the capability	**The hirer**, from its own edge: every connector in this profile is fed from the hirer's side or read with a hirer-issued credential, so the hirer can log what left and compare it with the connector manifest and `data_leaving` in the terms
What the capability reported, asked and was told	**The hirer**, at its hub: reports, approvals, outcomes and statements all pass through the hub, which holds them
That the receipts were not altered after the fact	**Anyone holding the terms**: the chain and the signatures, with `receipt_head` checkpoints at the hub, as in every profile
What the capability did with the hirer's data at the provider's, including what it sent to a model	**Only the capability says so.** The receipts record it, under the capability's signature; nothing on the hirer's side can confirm it. The terms (`data_leaving`, `model`) are the promise, and the receipts are the provider's account of keeping it
What was deleted at the provider's when the hire ended	**Only the capability says so**, in its `hire_ended` receipt (section 8, rule 5)

Rules:

1. The terms of a Hosted hire MUST say in `data_leaving` what the hirer's data is used for at the provider's, including every onward recipient (a model provider, a sub-processor) by name, and in `model` which model is used and where it runs.
2. A Hosted capability MUST write a receipt of kind `hosted/data_received` for each batch or stream it takes in (the connector, the counts, never the content), and `hosted/data_sent_on` for each onward transfer to a model or any other recipient, so that the account is at least complete.
3. A hub, a registry or a document that shows a Hosted hire's track record SHOULD show which of its evidence is hirer-checkable and which is the capability's own account, in those words or their equivalent.
4. Nothing in this profile makes the capability's own account verifiable. Appendix A names where that could be added by extension; until then a hirer chooses this profile knowing the split.

Rationale. The common case has to be covered, and it has to be covered without a claim it cannot keep. A hirer that reads this table knows exactly what it is trusting the provider for, which is more than any hosted service tells it today, and the receipts give the provider a way to be audited on its promises rather than merely to make them.

4. The four rules of hosting

This profile adds two receipt kinds (section 3) and no states or problem types. A capability conforms to this profile when it keeps all four rules:

1. **Outbound only.** It calls the hirer's hub; nothing at the hirer's is called by it except the connectors the terms name, and nothing at the provider's listens for the hub.
2. **The hub stays at the hirer's.** The capability MUST NOT act as, host, or proxy the hub for the hire it is working under.
3. **One tenant's data, one tenant's hire.** State, receipts and the signing key of one hire MUST be kept separate from every other hire's at the provider's, and MUST be deleted or returned when the hire ends, as the terms say.

4. **Declared up front.** Every onward recipient of the hirer's data is named in the terms before the hire starts (section 3, rule 1), and the daily limit, the autonomy ceiling and `dry_run` apply exactly as in the core.

5. Connectors, records and declared actions

The connector manifest and the record are those of the Resident profile, sections 4.1 to 4.3, with two differences of meaning. `data_location` says where the data is read *from* and that it is copied to the provider's; `auth` names the hirer-issued credential by label (section 6). The connector kinds that fit this profile are those fed from the hirer's side, for example `push` (the hirer sends), `export` (the capability fetches a hirer-issued export), `api` (the capability reads a hirer API with a scoped token), and `mcp` (the capability reads a hirer-operated MCP server).

Declared actions take the form of the Resident profile, section 5: an argument list run without a shell, `{service}` filled only from `services`. In this profile the command runs at the provider's and reaches the hirer's systems only through a connector with `act` scope and the acting credential of section 6. A command that would run inside the hirer's environment is not possible in this profile; a job that needs one is a Resident hire.

6. Credentials

The core's rules for credentials (section 9.4) apply, with these additions for a capability the hirer does not run:

1. Every credential a Hosted capability holds for a hirer's connector or action MUST be issued by the hirer, MUST be short-lived (24 hours RECOMMENDED, renewed through the hirer's own means), and MUST be revocable by the hirer at once.
2. The acting credential MUST be separate from every reading credential and MUST be scoped to the declared actions' targets only.
3. The capability MUST NOT store a credential beyond its lifetime, and MUST NOT reuse a credential of one hire for another, even for the same hirer.
4. The terms list every credential by label and scope in `credentials`, as the core requires; the receipts name the label on every use.

Example. `credentials: [{"label": "timesheets-export", "scope": "connector time-entries, read, renewed daily"}, {"label": "reminder-sender", "scope": "action send-reminder, hirer mail relay"}]`.

7. The HTTP binding

The seven calls, their paths, their messages and their examples are those of the Resident profile, section 6, without change. The profile is named at enrolment: `profile: "hosted"`. A hub that supports both profiles lists both in `supported` when it refuses an unknown one.

8. Operating rules

1. There MUST be only one running instance per hire, on one state store.
2. After a hire ends, the capability MUST stay idle (or exit) for that hire and MUST NOT enrol again unless given a new enrolment token.

3. Heartbeats SHOULD be sent at least every 5 minutes; the inbox SHOULD be collected every `poll_seconds`.
4. The retry, unreachable-hub and refused-credential rules of the core apply to every call. When the hub is unreachable the capability MUST stop taking in new data from the hirer's connectors after its retries are spent, and write `hub_unreachable`; it MAY keep what it already holds, under the terms.
5. At the end of the hire the capability MUST delete the hirer's data and the hire's state at the provider's, or return them, as the terms say, and MUST say so in the `hire_ended` receipt's `body` (`data_disposed`, with what and when). The receipts file itself is kept and SHOULD be given to the hirer.
6. A handed-off item in the inbox (core, section 6.5) is untrusted input and is taken up under the capability's own ref with `origin`.
7. On receiving an attestation, the capability SHOULD verify it and write `attestation_received`.

9. Conformance

A capability conforms to this profile when it conforms to the core, keeps the four rules of section 4, carries connector manifests and `credentials` in its terms, writes the two receipt kinds of section 3, runs declared actions as section 5 says, and makes the seven calls with messages that validate against their schemas. A hub conforms when it answers those seven calls as the Resident profile's section 6 and the core say, and accepts `profile: "hosted"` at enrolment. A conformance claim names the version and the profile: "SaaCIP 0.2, Hosted profile, capability".

Appendix A. Where attestation of the provider's environment would plug in

The capability's own account of what it did with the hirer's data (section 3) could become checkable in two known ways, neither of which this version defines. A **model provider's signed usage record**, if model providers publish one, could be referenced from each `hosted/data_sent_on` receipt as an `external_receipt` (core, section 14), tying the capability's claim of what it sent to the model to the model provider's record of what it received. An **attestation from a confidential computing environment** could be carried in the terms as an extension, so that the hirer knows the code and the environment it ran in before the hire starts. Both are extension points: an `ext` member on the terms, and `external_receipt` on receipts. This document names them so that implementers put them in the same place.

Appendix B. Change log

Version	Date	Changes
0.2 draft	6 November 2026	First version of the Hosted profile: the evidence split, the four rules of hosting, connectors fed from the hirer's side, short-lived hirer-issued credentials, two receipt kinds, data disposal at the end, the binding shared with the Resident profile, and the extension points for attestation of the provider's environment

Comments are welcome. The views here are the author's own.

Important notice. This specification is part of my ongoing research into the Agentic Economy and reflects my thinking at the time of writing. It is shared in good faith, for information and discussion. It is not legal, regulatory, technical, financial or other professional advice, and it is not a verdict on how the EU AI Act or any other law applies to any organisation, system or situation. Laws, guidance, technology and my own research all move on, and others may reach different conclusions. The examples, designs, diagrams and technology choices show one possible approach. Before relying on, adapting or applying any part of it in your own situation, you must test and verify it, design it for your own architecture, and take independent expert advice, including from qualified lawyers and from

your own IT, security and data protection specialists. Any decision to act on it, and its consequences, rest with you. To the fullest extent permitted by law, the author accepts no responsibility or liability for any loss, damage or consequence arising from any use of, or reliance on, this specification, which is provided without any warranty of accuracy, completeness or fitness for a particular purpose. Reading it does not create a professional or advisory relationship with the author. The views expressed are the author's own.