PROTOCOLO · VERSÃO 0.1 · RASCUNHO PARA DISCUSSÃO

MICP

Model Integration Context Protocol
Integrações com credenciais geridas para sistemas autónomos

25 de novembro de 2026
Especificação técnica · 10 páginas
Estado: proposta aberta · implementação de referência em curso

RESUMO

Os agentes de software já conseguem chamar ferramentas, mas continuam sem uma forma segura e portátil de usar integrações que exigem credenciais. Na prática, as chaves acabam em variáveis de ambiente, em ficheiros de configuração ou — pior — no contexto do modelo. Este documento propõe o **MICP**, um protocolo declarativo para descrever, provisionar, autorizar, executar e auditar integrações. Uma integração é declarada num pack que contém os passos para obter a credencial, as operações que expõe com os respectivos contratos de entrada e saída, e as regras de governação. O pack não contém segredos: contém referências a um cofre, cujo conteúdo é decifrado apenas no instante da chamada, no servidor. O resultado é uma integração que qualquer agente — ou qualquer programa determinístico — pode usar sem nunca ver uma chave.

1 O problema

Dar a um agente acesso a um serviço externo parece resolvido: existem bibliotecas, conectores e protocolos de chamada de ferramentas. O que continua por resolver é tudo o que vem antes e depois da chamada.

FALHA ACTUAL	CONSEQUÊNCIA
A chave vive numa variável de ambiente ou num ficheiro de configuração	Qualquer código no mesmo processo a lê. Num sandbox, `printenv` chega
A chave é colada no prompt «só para testar»	Fica no histórico, no fornecedor do modelo e nos registos — para sempre
Não existe modelo de autorização	Um agente com acesso ao Stripe pode reembolsar; ninguém definiu que não devia
Não existe auditoria por chamada	«O que é que os vossos agentes fizeram com as minhas credenciais?» não tem resposta
Não existe idempotência	Uma retentativa manda o email duas vezes, ou cobra duas vezes
A configuração é prosa em documentação	Cada utilizador interpreta os passos à sua maneira; o suporte repete-se para sempre

Nenhuma destas falhas se resolve com mais um conector. Resolvem-se com **contrato**: uma forma declarada de dizer o que a integração aceita, o que devolve, que segredo usa, quem o pode usar, e como se prova que funciona.

2 Princípios de desenho

P1. O segredo nunca entra no contexto. Nem no prompt, nem no resultado de uma ferramenta, nem nos registos. O modelo vê referências, nunca valores.

P2. O segredo nunca entra na sandbox. Onde corre código arbitrário não há credenciais. O comando pede, o servidor executa.

P3. Declaração em vez de código. Uma integração é um documento com contrato, não um programa. É o que permite validar, versionar, comparar e — sobretudo — recusar.

P4. Execução determinística. A mesma entrada, na mesma versão, produz o mesmo pedido. O modelo decide o quê; o protocolo decide como. Sem esta separação não há auditoria possível: só narrativa.

P5. Descoberta filtrada por autorização. Um agente não deve saber que existe uma operação que não pode usar. A lista de ferramentas é derivada, por chamador, no início de cada execução.

P6. O provisionamento faz parte do protocolo. Obter a credencial é metade do trabalho real — e é a metade que ninguém especifica. Os passos, os links, as esperas e as validações pertencem ao pack.

> **FRONTEIRA** O MICP não define transporte nem serialização de chamadas de ferramentas. Define o que uma integração é: como se obtém a credencial, o que aceita, o que devolve, quem a pode usar, o que fica registado e como se evita duplicar efeitos. Adaptadores (MCP, REST, CLI) expõem-na — não a definem.

3 Arquitectura

Um **pack** descreve uma integração. A sua execução assenta em cinco secções, cada uma com uma responsabilidade única.

SECÇÃO	RESPONSABILIDADE	DETERMINÍSTICA
`provisionamento`	Como obter a credencial: passos, variantes, validações	uma vez
`celulas`	Contrato de cada operação: entrada, saída, mapeamento, transporte	sim
`cofre`	Referências a segredos; envelope de cifra; rotação	sim
`governacao`	Autorização por célula, auditoria, idempotência, destinos permitidos	sim
`descoberta`	O que cada chamador vê, filtrado pelo que lhe foi concedido	sim

A unidade de execução chama-se **célula**. Uma célula é uma operação única — `email.enviar`, `stripe.cliente_criar` — com uma versão própria. Uma célula pode ser descrita numa frase:

```
Uma célula é uma função pura cujo ambiente contém o segredo.

  entrada tipada → validação → resolução do segredo → pedido determinístico
                 → validação da resposta → auditoria → saída tipada
```

A opção por uma célula por operação não é estética. É o que permite conceder `email.enviar` sem conceder `email.listar`, versionar uma operação sem tocar nas outras, e testar cada contrato em isolamento.

4 O pack

Um pack é um ficheiro JSON com a extensão `.micp.json`. Todos os objectos são fechados (`additionalProperties: false`): um protocolo não é um saco de configuração.

```
{
  "micp": "0.1",
  "integracao": {
    "apelido": "email_elizeu",
    "descricao_modelo": "Conta de ENVIO do Eliseu. Usa quando o pedido disser 'o email do
Eliseu'.",
    "auth": { "esquema": "basic", "segredo": "$cofre:email_elizeu.app_password" },
    "config": { "email": "elizeu@exemplo.pt" },
    "destinos_permitidos": ["smtp.gmail.com", "accounts.google.com"]
  },
  "provisionamento": { "variantes": [ /* §5 */ ] },
  "celulas": [ /* §6 */ ]
}
```

CAMPO	REGRA
`apelido`	Estável, único por workspace; é o que dá nome à ferramenta
`descricao_modelo`	≤ 300 caracteres, delimitada como dado, nunca como instrução
`auth.segredo`	Obrigatoriamente `$cofre:nome.campo`. Qualquer valor que pareça uma chave real é erro de validação
`config`	Apenas valores não secretos (endereço, conta, região)
`destinos_permitidos`	Allowlist obrigatória de domínios. O runtime recusa qualquer outro

5 Provisionamento

Obter a credencial é um procedimento com falhas previsíveis: um serviço pede a criação de uma aplicação numa consola, outro exige uma conta empresarial, outro demora dois minutos a propagar uma permissão. O protocolo declara esse procedimento para que possa ser **executado e validado**, em vez de lido e interpretado.

TIPO DE PASSO	QUEM AGE	O QUE PRODUZ
`abrir`	humano, guiado	um valor colado (chave, id, email)
`recolher`	humano	um valor simples, validado no local
`criar_automatico`	executor	uma credencial criada por API, escrita directamente no cofre
`esperar`	executor	nada — aguarda e verifica propagação

O mesmo objectivo pode ter **variantes** — e a escolha é do utilizador, não do protocolo:

```
"variantes": [
  { "id": "app_password",
    "nome": "Mais simples · ~5 min · conta pessoal",
    "passos": [
      { "n": 1, "acao": "abrir", "url": "https://myaccount.google.com/security",
        "instrucao": "Activa a verificação em duas etapas", "confirmar": true },
      { "n": 2, "acao": "abrir", "url": "https://myaccount.google.com/apppasswords",
        "instrucao": "Cria uma palavra-passe de aplicação chamada 'pipeline'",
        "obtem": "app_password", "segredo": true,
        "validar": "^[a-z]{4} [a-z]{4} [a-z]{4} [a-z]{4}$",
        "se_falhar": "Esta conta tem palavras-passe de aplicação desactivadas. Usa a variante
service_account." },
      { "n": 3, "acao": "recolher", "campo": "email", "obtem": "email" } ] },

  { "id": "service_account",
    "nome": "Consola Google · ~20 min · automação",
    "passos": [
      { "n": 1, "acao": "abrir", "url": "https://console.cloud.google.com/projectcreate",
        "instrucao": "Cria um projecto", "obtem": "project_id" },
      { "n": 2, "acao": "abrir", "url": "https://console.cloud.google.com/apis/library/
gmail.googleapis.com",
        "instrucao": "Activa a Gmail API",
        "espera": "120s", "verificar": "GET /v1/projects/{project_id}/services/
gmail.googleapis.com" },
      { "n": 3, "acao": "criar_automatico",
        "api": "iam.googleapis.com/projects/{project_id}/serviceAccounts",
        "instrucao": "O executor cria a conta de serviço", "obtem": "conta_de_servico" },
      { "n": 4, "acao": "criar_automatico",
        "api": "iam.googleapis.com/.../serviceAccounts/{conta}/keys",
        "instrucao": "O executor cria a chave JSON e guarda-a no cofre", "obtem":
"chave_json", "segredo": true } ] } ]
```

PROGRESSO O estado do provisionamento é persistido: variante escolhida, passo corrente e valores já obtidos. Um procedimento de vinte minutos pode ser interrompido e retomado dias depois, sem recomeçar. Sem esta peça, os provisionamentos morrem sempre a meio — e com eles a confiança no produto.

6 Células

Uma célula declara o contrato de uma operação. O contrato de entrada não é decoração: é a fronteira entre «um pedido legítimo» e «um efeito indesejado». Um campo com regra recusa um valor absurdo antes de existir qualquer chamada externa.

```
{ "id": "telegram_suporte.enviar", "versao": 1, "escopo": "escrita", "transporte": "http",
  "descricao_modelo": "Envia uma mensagem de texto. Usa para avisar que algo terminou ou
falhou.",
  "input_schema": {
    "texto":   { "tipo": "texto",   "obrigatorio": true, "min": 1, "max": 4096 },
    "chat_id": { "tipo": "texto",   "obrigatorio": false, "omissao":
"{{config.chat_id_padrao}}" },
    "silencioso": { "tipo": "booleano", "omissao": false } },
  "mapeamento": {
    "metodo": "POST",
    "url": "https://api.telegram.org/bot{{segredo.bot_token}}/sendMessage",
    "corpo": { "chat_id": "{{input.chat_id}}", "text": "{{input.texto}}",
               "disable_notification": "{{input.silencioso}}" } },
  "output_schema": { "ok": { "tipo": "booleano" }, "id_mensagem": { "tipo": "numero" } },
  "verificar": { "metodo": "GET", "url": "https://api.telegram.org/bot{{segredo.bot_token}}/
getMe" } }
```

O mapeamento admite exclusivamente três origens de variáveis: `{{input.*}}`, `{{config.*}}` e `{{segredo.*}}`. Qualquer outra é erro de validação — é assim que se impede que um pack invente caminhos para dados que não devia alcançar.

TRANSPORTE	CAMPOS EXIGIDOS	NOTA
`http`	`metodo`, `url`, `corpo`	JSON; binários por referência
`smtp`	`servidor`, `porta`, `tls`	envio de correio
`imap`	`servidor`, `porta`, `tls`	leitura de correio

Três transportes, e nenhum outro. A tentação de permitir código arbitrário dentro da célula destrói as propriedades que justificam o protocolo: um executor capaz de correr shell com o segredo no ambiente é, na prática, um serviço com acesso a credenciais — e o `printenv` volta pela porta do lado.

7 Cofre

O pack não contém segredos. Contém **endereços**. É esta separação que permite que um pack circule entre pessoas e organizações sem transportar credenciais: quem recebe o ficheiro recebe a receita, nunca a chave.

O cofre guarda o valor cifrado com uma chave de dados por segredo; a chave de dados é cifrada com a **chave primária**, que vive fora da base de dados.

```
segredo        → AES-GCM com chave de dados (por segredo)
chave de dados → cifrada com a chave primária (versão em key_version)
chave primária → ficheiro no servidor · backup cifrado, guardado fora do servidor
```

> **DUAS CONSEQUÊNCIAS QUE SE PAGAM CARO QUANDO IGNORADAS** A chave primária não se cifra a si mesma: se o backup da base de dados viajar sozinho, todas as linhas do cofre são irrecuperáveis. O backup da chave tem de existir separado, e o restauro tem de ser ensaiado — um cofre que nunca foi aberto a partir do backup é uma promessa, não um mecanismo.

A chave fica no servidor, não no utilizador, por uma razão funcional: às três da manhã, quando uma rotina corre sozinha, ninguém está acordado para desbloquear nada. O utilizador conserva o que importa para a confiança — revogar, restringir, ver quem usou e quando — sem ser obrigado a estar presente para a automação funcionar.

A decifra ocorre uma vez por chamada, em memória, e o valor é descartado no fim. Nunca é escrito em disco, em registo, em eventos de execução ou no contexto do modelo. A rotação usa `key_version`: as linhas antigas continuam a abrir com a versão anterior e são recifradas à medida que são usadas.

8 Governação

Autorização por célula. A concessão (grant) refere uma célula concreta, não uma integração inteira. «Este agente pode enviar email; não pode listar a caixa de entrada» é uma frase que o sistema compreende.

Escopo por omissão permissivo, restrição por excepção. Ao ligar uma integração, ela fica disponível a todo o workspace. Restringir a um agente é uma acção explícita. A fricção fica onde é rara e não onde é diária — com a consequência assumida de que a protecção efectiva depende da auditoria e do escopo de leitura/escrita declarado em cada célula.

AMEAÇA	MITIGAÇÃO NO PROTOCOLO
Pack malicioso exfiltra a chave	Allowlist de destinos obrigatória; hosts visíveis antes de abrir; selo de verificação
Instruções escondidas numa descrição	`descricao_modelo` limitada, delimitada como dado, sem efeito sobre comportamento
Código na sandbox imprime o ambiente	P2: a sandbox não recebe credenciais; recebe um passe de execução de curta duração
Retentativa duplica o efeito	`chave_idempotencia` obrigatória por execução; a segunda devolve o resultado da primeira
Replay de um pedido assinado	Assinatura com carimbo temporal; janela de aceitação curta
Agente usa operação que não devia	Descoberta filtrada: a célula não aparece no catálogo do agente

Auditoria. Cada execução produz um registo com: célula e versão, apelido da integração, agente, execução e passo, resultado, código de erro, duração e carimbo temporal. Nunca o segredo, nunca o corpo completo de um pedido que o possa conter.

Idempotência. A chave de idempotência é parte do contrato de execução, não uma opção. Sem ela, cada retoma de uma execução interrompida é uma oportunidade de enviar dois emails, criar dois clientes ou cobrar duas vezes.

9 Descoberta

CHAMADA	DEVOLVE
`GET /micp/manifesto`	Versão do protocolo e transportes suportados
`GET /micp/celulas`	Células visíveis **para aquele chamador**: identificador, versão, descrição, contratos, escopo
`GET /micp/celulas/{id}`	Detalhe de uma célula e exemplo de entrada

A listagem é **filtrada pela autorização**. Uma célula sem concessão não é recusada em execução: não é listada. O agente não sabe que existe. Isto reduz o risco de tentativa cega e mantém o contexto pequeno, o que importa quando um workspace tem dezenas de integrações.

Como a lista de ferramentas é derivada no início de cada execução, ligar uma integração produz efeito imediato: o agente seguinte já a tem. Não há implantação, nem reinício, nem alteração de código.

10 Erros

Erros são valores, não excepções genéricas. Um executor distingue «a entrada está mal» de «o fornecedor está em baixo» — e a diferença decide se se tenta outra vez, se se corrige o pedido ou se se acorda alguém.

CÓDIGO	SIGNIFICADO	RETENTÁVEL
`input_invalido{campo}`	Falhou o contrato de entrada	não
`nao_autorizado`	Sem concessão para esta célula	não
`cofre_indisponivel`	Chave primária ausente ou versão desconhecida	não
`destino_nao_permitido`	Host fora da allowlist	não
`falha_fornecedor{status}`	O serviço externo recusou ou falhou	conforme o estado
`conflito_idempotencia`	Chave repetida com entrada diferente	não
`versao_nao_suportada`	Célula pedida numa versão inexistente	não

11 Conformidade

Um protocolo sem testes de conformidade é uma opinião. Uma implementação diz-se conforme quando passa os doze pontos seguintes, verificação automática incluída.

1. `micp` presente e igual a `0.1`
2. Campos obrigatórios da integração presentes; `descricao_modelo` ≤ 300 caracteres
3. `auth.segredo` casa `$cofre:` e não parece um segredo real
4. Identificador de célula estável (`recurso.acao`) e versão inteira ≥ 1
5. `input_schema` não vazio, com tipos conhecidos
6. `output_schema` presente e não vazio
7. Mapeamento usa apenas `{{input}}` , `{{config}}` , `{{segredo}}`
8. Transporte válido e coerente com o mapeamento
9. Allowlist presente e todos os hosts do pack estão nela
10. `verificar` presente — integração sem teste não conforma
11. Passos de provisionamento coerentes: cada valor obtido é usado; variantes frágeis declaram `se_falhar`
12. Nenhum valor do pack contém padrões de segredo conhecidos (`sk_live_` , `AKIA` , `-----BEGIN` , …)

12 Comparação

	CHAMADA DE FERRAMENTAS	CONTRATO DE API	COMO SE OBTÉM A CREDENCIAL	AUTORIZAÇÃO POR AGENTE	AUDITORIA POR CHAMADA	IDEMPOTÊNCIA
MCP	sim	por ferramenta	—	—	—	—
OpenAPI	—	sim	—	—	—	—
n8n / Zapier	via nó	por nó	documentação	—	parcial	—
MICP	por adaptador	por célula	**parte do protocolo**	**por célula**	**obrigatória**	**obrigatória**

O MICP não substitui nenhum destes: acrescenta-lhes a camada que falta. Uma integração descrita em MICP pode ser exposta por MCP, por REST ou por uma linha de comandos — o que se mantém constante é o contrato, a autorização e o registo.

13 Limitações e trabalho futuro

- **Três transportes.** Serviços que exigem binários multipart, WebSocket ou protocolos proprietários ficam de fora até existir um transporte dedicado.
- **Provisionamento parcialmente manual.** Muitos fornecedores não expõem API para criar credenciais; nesses casos o protocolo guia e valida, mas não substitui o humano.
- **Chave primária centralizada.** Um modo com chave por utilizador é possível, mas incompatível com execução autónoma — fica como variante organizacional, não como omissão.
- **Versões de célula.** A semântica de convivência entre versões (depreciação, migração assistida) está definida em intenção, não em experiência.
- **Confiança entre organizações.** O selo de verificação depende de quem assina. Um modelo de reputação distribuída está fora do âmbito da versão 0.1.

14 Implementação de referência

A especificação foi escrita a partir de um sistema em produção, e não o contrário — o que permite distinguir com precisão o que já existe do que é intenção.

ELEMENTO	ESTADO
Manifesto de integração, operações `recurso × ação`	implementado
Contrato de entrada/saída, verificação obrigatória	implementado
Concessões por agente, auditoria por chamada	implementado
Descoberta filtrada por concessão	implementado
Importação a partir de OpenAPI	implementado
Cofre com envelope e chave primária versionada	em curso
Provisionamento com progresso persistido	em curso
Idempotência por execução	**v0.2**
Adaptadores MCP, REST e CLI	**v0.2**

A Anexo — adaptadores

Um adaptador traduz o contrato para um transporte. Não acrescenta semântica; se acrescentasse, deixaria de ser adaptador.

ADAPTADOR	MAPEAMENTO	USO
MCP	célula → ferramenta; `input_schema` → schema de argumentos	clientes de agentes existentes
REST	`POST /celulas/{id}/executar` com `{input, chave_idempotencia}`	integração directa em aplicações
CLI	`pipeline-celula email.enviar --para …`	passos de shell, com passe de execução de curta duração

> **PROPRIEDADE QUE O PROTOCOLO PROMETE** A mesma célula, chamada por três adaptadores diferentes, produz o mesmo pedido, deixa o mesmo registo e nunca expõe o segredo. É esta repetibilidade — e não a elegância do formato — que torna a integração auditável.

MICP · Model Integration Context Protocol · v0.1 · 25 de novembro de 2026 · Documento para discussão. Comentários e propostas de alteração são bem-vindos antes da publicação da versão 0.1.