MachineTransfer

Agent payments. On your terms.

MachineTransfer is a protocol project for purchasing useful digital services through AI agents. Its ambition is a common transaction language across agent workflows: a clearly defined result, an exact payment authorization and a verifiable receipt. The agent helps find and prepare the purchase. The operator decides what is allowed. The wallet authorizes the payment. The merchant checks what arrived.

This separation makes the transaction understandable to both people and software. A request for a data report, a model inference or a tool result becomes a bounded order with a named service, a recipient, a price and a validity window. Payment can then be checked against those terms rather than inferred from a chat message or a successful API response.

MTR is the payment asset specified by the protocol's Ethereum ERC20 architecture. It combines a one-billion-token genesis supply, a fixed 0.85% tax on gross transfers and 5% annual issuance to an immutable treasury. Those rules belong to the token contract; operator policy and merchant fulfillment belong to separate application layers.

Why choose MTR

MTR's adoption proposition combines useful services that accept it, competitive complete pricing, reliable fulfillment, readable receipts and a standard operator-controlled integration. The test is concrete: does this route buy the same useful result on better overall terms? MachineTransfer makes that comparison explicit while keeping payment authority with the owner.

Useful work, clearly priced

Agents often need information or capabilities beyond the model itself. A research assistant may need a fresh dataset. A software agent may need a specialized analysis. A business workflow may need a document converted into a structured result. These are natural units for a purchase when the seller can describe the output precisely.

Service concept	A bounded deliverable
Data access	One named dataset or report with an observation time
Model inference	One specified inference result with defined inputs
Specialist tools	One conversion, calculation or analysis artifact
API access	One result bundle tied to an invoice

The service catalog describes the result before the payment route. Two routes are comparable only when they purchase the same deliverable. A merchant can offer an MTR price that reflects genuine service economics, but the buyer compares that price after tax, gas and conversion. Acceptance, output quality, timing and repeatable delivery matter alongside the price.

A purchase with a finish line

A bounded order has a clear completion condition. The merchant defines what the buyer receives and the required net payment. The buyer evaluates the full cost. The wallet approves the exact transfer. The merchant records acceptance and returns the result associated with that invoice.

This approach gives both sides something concrete to inspect. The operator can explain why the agent selected the service. The merchant can identify which order a payment satisfies. A retry can refer to the same invoice and result instead of silently creating another purchase.

Quote. Authorize. Verify.

MachineTransfer separates the purchase into distinct decisions. Each decision has its own inputs and responsibility, so a model suggestion does not become wallet authority merely by moving through a workflow.

Quote

The purchase begins with service terms: what is delivered, who receives payment, which asset and chain apply, when the quote expires and what the merchant must receive after tax. A merchant-scoped registry identifies the service recipient and quote signer. The merchant's signed quote binds that current service revision, the buyer, request, token, gross price, minimum net receipt, validity window and nonce. The operator still supplies approved routes and spending limits to the planner.

Authorize

The wallet reviews the exact payment. The signed transfer binds the sender, recipient, gross amount, validity interval and unique nonce to the token contract and chain. Changing those terms produces a different authorization. A relay can submit an authorized transfer without receiving a general allowance to spend the holder's balance.

The contract consumes the nonce when executing the transfer. Used or cancelled nonces cannot authorize a second transfer. The holder can cancel an unused nonce on chain. Expiry and replay protection keep the authorization specific to its intended transaction.

Verify

Service settlement checks the current registry entry and merchant quote before executing the buyer's exact signed transfer. The merchant checks both the service-settlement event and token transfer against the approved order, including sender, recipient, nonce and net amount. Receipt checking uses canonical, finalized observations from a trusted chain reader.

Deliver

The merchant reference ledger durably binds an invoice to its payment identity and a local access record. Its recovery path preserves that association across retries and interrupted writes. An external service connects its own result storage and idempotent delivery to that record, keeping settlement and fulfillment as distinct responsibilities.

A protocol with defined boundaries

MachineTransfer's architecture separates the asset contract from the software that prepares purchases and the merchant system that delivers them. This makes each component easier to reason about and lets developers place authority where it belongs.

Layer	Responsibility
MTR contract	Token balances, transfer tax, issuance and signed-transfer execution
Service registry and quotes	Service identity, recipient and authenticated terms
Service settlement	Current quote validation and a service event linked to the token transfer
Receipt checking	Match service and token settlement against trusted chain data
Merchant application	Durable invoice claims, replay handling and result delivery

Ethereum and wallet standards

Ethereum mainnet is MTR's canonical launch chain; ETH pays gas. The 18-decimal ERC20 design supports signed allowance approvals, exact EIP-712 transfers and ERC-1271 validation for compatible contract wallets.

The scaling direction is an Ethereum-settled EVM rollup, with infrastructure and bridge design still to be selected. One canonical MTR supply remains on Ethereum. Any representation must conserve it, without a second genesis or issuance stream.

Token policy is fixed: no administrative upgrades, tax changes, arbitrary minting or account freezes. The immutable treasury controls its own assets, not other holders' balances.

Batching groups individually authorized transfers into one atomic execution. Each retains its recipient, amount, nonce and validity. Shared overhead can reduce per-payment cost; the submitting party still pays Ethereum gas.

MTR token policy

MTR is a variable-price asset used in the protocol design. Its value in another currency depends on the market and the applicable conversion quote. The contract does not set a fiat price or peg. Service pricing therefore distinguishes the token amount from the value and cost of the purchased result.

Fixed transfer tax

Every token transfer applies a tax of 85 basis points, or 0.85%, to the gross amount. In token base units, the calculation is `tax = floor(gross × 85 / 10,000)` and `net = gross - tax`. The tax goes to the immutable treasury. There are no discretionary address exemptions, and minting is untaxed.

Example transfer	MTR amount
Gross authorized amount	100.00
Treasury tax	0.85
Recipient's net transfer leg	99.15

The signed amount and any allowance are gross. A merchant's requested net amount needs a corresponding gross quote calculated with exact base-unit rounding. Multiple transfer legs each apply their own tax; their combined effect is accounted for across the whole route.

Supply and treasury issuance

The contract specifies 1,000,000,000 MTR at genesis. Completed 365-day epochs add 5% of the preceding scheduled supply, rounded down in base units. Anyone can settle due epochs, and every issuance goes to the immutable treasury. Delayed settlement processes completed epochs in bounded calls and compounds each epoch's increase. This is treasury issuance, not interest or yield paid to holders.

The complete cost

ETH pays Ethereum gas separately from MTR. Compare service price, every transfer tax, gas, acquisition, conversion, cash-out and attributable failure or support costs. Unknown costs stay explicit. The discount-threshold calculation identifies the modeled MTR price needed to match or beat the same accepted USDT deliverable. A tie means equal modeled cost; actual quotes and successful delivery establish purchase value.

Model integrations, operator control

MachineTransfer's local MCP planning server communicates over standard input/output using protocol version 2025-11-25. It exposes `list_services` and `plan_purchase`, with the catalog and policy supplied by trusted operator configuration at startup. OpenAI function-tool definitions expose the same bounded planning logic to an application's tool handler. Compatible integrations share the purchase vocabulary while preserving operator approval.

A narrow model input

The model selects a service identifier from the operator's catalog. It does not supply a new destination wallet or invent payment terms. The trusted controller provides the accepted assets, token identities, recipients, service allowlist, quote observations, freshness limits and maximum complete cost.

The planner evaluates those supplied quotes for the selected deliverable. It excludes routes outside policy, expired quotes and routes with unknown costs. Eligible routes are compared on modeled complete cost. Equal-cost alternatives require review rather than an arbitrary automatic selection.

Planning and execution

The planner is an offline component: it evaluates supplied data and returns a purchase plan for review. Quote fetching, merchant authentication, wallet signing and transaction submission are separate integration responsibilities. This separation gives a model useful purchasing context while preserving the operator's control over money movement.

A signed authorization is the execution boundary. The model's explanation is useful context, but the wallet approves the concrete fields that the contract accepts. Receipt checking then evaluates the resulting transaction independently of that explanation.

Developers can use this pattern inside a research assistant, a coding agent or a business workflow. The common contract is the same: make the service understandable, keep the payment exact and preserve a clear record of the result purchased.

Building with MachineTransfer

An integration starts with a specific service and a clear deliverable. The merchant determines its required net amount and quote lifetime. The operator chooses permitted routes and a complete-cost limit. Payment preparation expresses the exact authorization, and the wallet reviews it before execution.

The merchant checks settlement using its approved order, the intended token identity and trusted Ethereum observations. Its invoice record claims the payment once and associates it with the result. Delivery uses the same invoice identity, so repeat requests can retrieve or resume that result without creating another service effect.

Growth through useful purchases

The adoption path starts with one service worth buying again. A merchant earns repeat purchases by delivering the promised result at a sustainable complete price. An operator reuses the integration when its cost, control and fulfillment fit the workflow. More useful accepted services give buyers more reasons to return; repeat purchases give merchants a reason to keep improving those services.

This growth depends on voluntary choices by service providers and agent operators. A shared integration makes the purchasing workflow portable across compatible tools and models, while each operator retains its own acceptance and spending policy.

A shared language for purchases

The design brings together three records: the order that describes the work, the authorization that approves the transfer and the receipt that describes settlement. Keeping these records consistent gives operators a readable transaction and developers a precise integration boundary.

MachineTransfer centers the relationship between useful agent services and explicit owner authority. The asset supplies consistent monetary rules. The application supplies service terms and cost information. The wallet supplies consent. The merchant supplies the result.

Explore the protocol

Read the payment workflow, the MTR token design and the developer overview for the corresponding protocol and integration details.