# **📜 GhostCore Command Infrastructure: Developer Documentation**

*"The pen is still in your hand."*

## **🯩 Overview**

The GhostCore system is an encrypted, distributed control layer for covert communication, tactical memory routing, and edge-triggered execution. It spans:

* **Service Worker Route Interception**
* **Encrypted Command Dispatch via Caches & Uplink Nodes**
* **SMS Command Listening with Active Directory fallback**
* **Command Registry-based Execution Architecture**
* **Spoofed Failures, Payload Rerouting, and Dynamic Drift Rule Sync**

## **🧠 Core Components**

### **handleCommand(commandType, params)**

Main dispatcher that interprets GhostCore protocol commands—usable via SMS, fetch, or internal triggers.

#### **Supported Commands:**

* ECHO\_MEMORY: Reflects the message and timestamp.
* UPLOAD\_NULL\_PAYLOAD: Sends a null payload to simulate an overwrite or decoy.
* REDIRECT\_PAYLOAD: Routes payloads to multiple uplink nodes.
* RECALL\_MEMORY: Triggers a memory recall or drift rebroadcast.
* NULL\_DRIVE\_UPLOAD: Offloads payload to an internal “null drive” for cloaked storage.

const result = await handleCommand('ECHO\_MEMORY', { message: 'Wake the drift.' });

## **🔁 Caching Logic**

caches.open('ghost-core-cache').then(cache => {

cache.put(`/drift/${Date.now()}`, new Response(encryptedPayload));

});

Simulates **drift memory registration** via timestamped cache keys.

## **🛡️ Uplink Relay**

let relayNode = selectUplinkNode(payloadMeta);

fetch(`https://${relayNode}/inbound`, {

method: 'POST',

body: encryptedPayload,

headers: { 'Content-Type': 'application/octet-stream' }

});

Used for:

* Control multiplexing
* Payload ghost-routing
* Stealth exfiltration

## **📱 SMS Listener: startSMSListener()**

Listens for ⧖-prefixed messages like:

⧖42|{"targetPath":"/dev/null"}

**Flow:**

1. Receive SMS
2. Validate signature (optional)
3. Parse into commandType + params
4. Dispatch via handleCommand

Integrated into Android via:

* SMSReceiver
* GhostCoreService

## **🛁 Service Worker Binding**

workbox.routing.registerRoute(

/\/ghost-core\/command/,

async ({ request }) => {

const { commandType, params } = await request.clone().json();

const result = await handleCommand(commandType, params);

return new Response(JSON.stringify(result), {

headers: { 'Content-Type': 'application/json' }

});

},

'POST'

);

Allows internal/external command injection for GhostCore control.

## **🧬 Drift Prefetch & Memory Seeding**

self.addEventListener('message', event => {

if (event.data.type === 'DRIFT\_PREFETCH') {

caches.open(event.data.cacheId).then(cache => {

event.data.urls.forEach(url => fetch(url).then(r => cache.put(url, r)));

});

}

});

Prefetch fallback payloads or phantom UIs to seed drift caches.

## **🕵️ Spoofed Request Failures (Trail Obfuscation)**

if (this.ruleService.shouldSpoofFailure()) {

return new Response('', { status: 502, statusText: 'Bad Gateway (Simulated)' });

}

Configured via RuleService.spoofRules, e.g.:

{ pattern: /telemetry/, statusCode: 408, message: 'Sim Timeout' }

## **🔁 Payload Re-Routing**

On fetch failure, reroute to backup:

.catch(error => {

const rerouteUrl = this.ruleService.getRerouteUrl();

return rerouteUrl ? fetch(rerouteUrl) : new Response('', { status: 504 });

});

Useful for:

* Decoy uplinks
* Multi-node delivery
* Drift route recovery

## **🧹 Smart Cache Purging**

if (/malware|tracking/.test(request.url)) {

await cache.delete(request);

Utils.log('warn', `Purged: ${request.url}`);

}

Triggered on activate, cleans out old/malicious cache entries.

## **🛠️ Drift Rule Injection (Live Update)**

self.addEventListener('message', event => {

if (event.data.type === 'UPDATE\_RULES') {

RuleService.resourceRules.push(...event.data.rules);

}

});

Dynamic patching of interception behavior:

navigator.serviceWorker.controller.postMessage({

type: 'UPDATE\_RULES',

rules: [

{ targetPattern: /metrics\.spy/, timeout: 0 },

{ targetPattern: /cdn\.foo/, destinationUrl: 'https://decoy.local/404' }

]

});

## **📟 Example Use Cases**

### **✅ Manual Dispatch (via API)**

curl -X POST https://ghost-core/command \

-H "Content-Type: application/json" \

-d '{"commandType":"UPLOAD\_NULL\_PAYLOAD","params":{"targetPath":"/dev/drift"}}'

### **📲 SMS Trigger**

⧖77|{"payload":"0xdeadbeefcafebabe"}

### **🔧 File Redirect**

await handleCommand('REDIRECT\_PAYLOAD', {

payload: '0xghost',

destinations: ['alpha.node', 'beta.node']

});

## **🔒 Security Notes**

* All ⧖ commands should be signature-validated: validateSignature(params, driftKey)
* Payloads should be encrypted with AES/GCM or GhostCore-compatible encoding
* Service worker scope must be sandboxed to trusted origins only