

**FRONTEND LLD**

***Version 1.1***

*November 2025*

**

|  |
| --- |
| **Revisions** |

| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| --- | --- | --- | --- |
| 1.1 | Siddharth Bhatia | Frontend LLD | 07-11-2025 |

|  |
| --- |
| **Review & Approval** |

**Requirements Document Approval History**

| **Approving Party** | **Version Approved** | **Signature** | **Date** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Requirements Document Review History**

| **Reviewer** | **Version Reviewed** | **Signature** | **Date** |
| --- | --- | --- | --- |
| **Jeevan Kumar** | 0.5 |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
|  |

Contents

No table of contents entries found.

# 

BPP Agent Low-Level Design (LLD) Document

Browser Protection Platform Agent (Windows & Linux)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Document Information

Version Date Author Description

1.0 2025-01-07 System Analysis Initial LLD Document

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Table of Contents

1. Introduction

2. System Overview

3. Technology Stack

4. Architecture Design

5. Module Design

6. Component Design

7. Platform-Specific Implementation

8. Security Implementation

9. API Integration

10. Extension Management

11. Service Management

12. License Verification

13. Error Handling & Recovery

14. Performance Optimization

15. Deployment Architecture

16. Appendix

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Introduction

1.1 Purpose

This document provides a comprehensive low-level design for the BPP (Browser Protection Platform) Agent, a cross-platform system service that manages browser extensions for phishing protection on Windows and Linux operating systems.

1.2 Scope

The BPP Agent is a Go-based system service that provides: - License verification and validation - Browser extension installation and management - System service management - Periodic health checks and updates - Cross-platform support (Windows & Linux) - GUI setup wizard (Windows only) - RESTful API server for browser communication

1.3 Audience

• Backend Developers

• System Administrators

• DevOps Engineers

• Security Engineers

• Technical Leads

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. System Overview

2.1 Application Type

Cross-platform system service/daemon written in Go

2.2 Supported Platforms

• Windows: Windows 10/11 (32-bit and 64-bit)

• Linux: Ubuntu, Debian, CentOS, RHEL (systemd-based distributions)

2.3 Key Features

2.3.1 Core Features

1. License Management

– MAC address-based license verification

– Periodic license validation

– Revocation handling

– Token-based authentication

2. Browser Extension Management

– Chrome extension installation

– Firefox extension installation

– Edge extension installation

– Dynamic extension configuration from backend

– Extension version tracking

3. System Service

– Windows Service (BPP\_AGENTService)

– Linux systemd service (bpp\_agent)

– Auto-start on boot

– Failure recovery mechanisms

4. API Server

– RESTful HTTP server on port 64321

– CORS-enabled for browser extensions

– Device information endpoint

– License status endpoint

5. GUI Setup Wizard (Windows Only)

– Step-by-step installation wizard

– Service setup automation

– License verification

– Extension installation

– System restart management

2.4 System Requirements

2.4.1 Windows

• Windows 10 or later

• Administrator privileges

• .NET Framework (for service management)

• Supported browsers: Chrome, Firefox, Edge

2.4.2 Linux

• systemd-based Linux distribution

• Root/sudo privileges

• dmidecode package

• Supported browsers: Chrome, Firefox, Edge

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Technology Stack

3.1 Core Technologies

Category Technology Version Purpose

Language Go (Golang) 1.18+ Core programming language

HTTP Router Gorilla Mux Latest HTTP routing

Windows API golang.org/x/sys/windows Latest Windows system calls

Encryption crypto/aes Built-in AES-256-CBC encryption

JSON encoding/json Built-in Data serialization

HTTP Client net/http Built-in API communication

3.2 External Dependencies

import (

"github.com/gorilla/mux" // HTTP routing

"golang.org/x/sys/windows" // Windows API (Windows only)

)

3.3 Build Configuration

Windows Build:

go build -ldflags="-H windowsgui" -o bpp\_agent.exe bpp.go

Linux Build:

go build -o bpp\_agent encrypt.go

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Architecture Design

4.1 High-Level Architecture

┌─────────────────────────────────────────────────────────────┐

│ BPP Agent Process │

│ ┌───────────────────────────────────────────────────────┐ │

│ │ Main Application │ │

│ │ ┌─────────────────────────────────────────────────┐ │ │

│ │ │ GUI Layer (Windows Only) │ │ │

│ │ │ - Setup Wizard │ │ │

│ │ │ - Progress Tracking │ │ │

│ │ │ - User Interaction │ │ │

│ │ └─────────────────────────────────────────────────┘ │ │

│ │ ┌─────────────────────────────────────────────────┐ │ │

│ │ │ Service Layer │ │ │

│ │ │ - License Verification │ │ │

│ │ │ - Extension Management │ │ │

│ │ │ - Periodic Health Checks │ │ │

│ │ └─────────────────────────────────────────────────┘ │ │

│ │ ┌─────────────────────────────────────────────────┐ │ │

│ │ │ API Server Layer │ │ │

│ │ │ - HTTP Server (Port 64321) │ │ │

│ │ │ - CORS Middleware │ │ │

│ │ │ - Device Info API │ │ │

│ │ │ - License Status API │ │ │

│ │ └─────────────────────────────────────────────────┘ │ │

│ │ ┌─────────────────────────────────────────────────┐ │ │

│ │ │ Platform Abstraction Layer │ │ │

│ │ │ - Windows-specific operations │ │ │

│ │ │ - Linux-specific operations │ │ │

│ │ │ - Browser detection │ │ │

│ │ │ - System information │ │ │

│ │ └─────────────────────────────────────────────────┘ │ │

│ └───────────────────────────────────────────────────────┘ │

└─────────────────────────────────────────────────────────────┘

↕

┌─────────────────────────────────────────────────────────────┐

│ Backend License Server │

│ (Django REST Framework) │

└─────────────────────────────────────────────────────────────┘

↕

┌─────────────────────────────────────────────────────────────┐

│ Browser Extensions │

│ (Chrome, Firefox, Edge) │

└─────────────────────────────────────────────────────────────┘

4.2 Component Interaction Flow

┌──────────────┐

│ Agent Startup│

└──────┬───────┘

│

▼

┌──────────────────┐

│ Initialize Logging│

└──────┬───────────┘

│

▼

┌──────────────────────┐

│ Check Admin/Root │

│ Privileges │

└──────┬───────────────┘

│

├─ Windows ──────────────┐

│ │

▼ ▼

┌──────────────────┐ ┌──────────────────┐

│ Show GUI Wizard │ │ Service Mode │

└──────┬───────────┘ └──────┬───────────┘

│ │

▼ │

┌──────────────────┐ │

│ Step 1: Service │ │

│ Setup │ │

└──────┬───────────┘ │

│ │

▼ │

┌──────────────────┐ │

│ Step 2: License │◄───────────┘

│ Verification │

└──────┬───────────┘

│

▼

┌──────────────────┐

│ Step 3: Extension│

│ Installation │

└──────┬───────────┘

│

▼

┌──────────────────┐

│ Start API Server │

└──────┬───────────┘

│

▼

┌──────────────────┐

│ Periodic Health │

│ Checks (2 min) │

└──────────────────┘

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Module Design

5.1 License Verification Module

5.1.1 Purpose

Validates device license with backend server using MAC address-based authentication.

5.1.2 Key Functions

checkLicenseValidity()

func checkLicenseValidity() LicenseStatus {

// 1. Collect device identifiers

deviceDetails := getDeviceIdentifiers()

// 2. Send to backend for verification

resp := POST(LICENSE\_VERIFICATION\_URL, deviceDetails)

// 3. Parse response

licenseResp := parseLicenseResponse(resp)

// 4. Update extension configuration from backend

updateExtensionsFromBackend(licenseResp)

// 5. Check revocation status

if isLicenseRevoked(licenseResp) {

deleteExtensions()

stopService()

exit()

}

return licenseStatus

}

5.1.3 Device Identifiers

Field Description Source

UUID System UUID dmidecode (Linux), WMI (Windows)

MacAddress Primary MAC address Network interfaces

SerialNumber System serial number dmidecode (Linux), WMI (Windows)

OSType Operating system type runtime.GOOS

OSPlatform OS platform runtime.GOOS

OSRelease OS release version uname -r (Linux), ver (Windows)

HostName Computer hostname os.Hostname()

Architecture CPU architecture runtime.GOARCH

CurrentVersion Agent version Hardcoded “1.0.0”

CurrentUser Logged-in user whoami, $USER

Browsers Installed browsers File system + registry checks

5.1.4 License Response Structure

type LicenseResponse struct {

STATUS string

Code int

Data struct {

ValidStatus bool

LicenseID string

ValidTill interface{}

Revoke interface{}

ValidFrom interface{}

AccessToken interface{}

RefreshToken interface{}

AgentVersion string

AllocationDate string

// Extension configuration

ChromeID string

ChromeURL string

EdgeID string

EdgeURL string

FirefoxID string

FirefoxURL string

// Plugin versions

ChromePluginVersion string

EdgePluginVersion string

FirefoxPluginVersion string

}

}

5.2 Extension Management Module

5.2.1 Purpose

Manages browser extension installation, updates, and removal across Chrome, Firefox, and Edge.

5.2.2 Extension Installation Flow

┌──────────────────────┐

│ Get Extension Config │

│ from Backend │

└──────┬───────────────┘

│

▼

┌──────────────────────┐

│ Kill Browser │

│ Processes │

└──────┬───────────────┘

│

▼

┌──────────────────────┐

│ Detect Browser │

│ Architecture │

└──────┬───────────────┘

│

├─ Chrome ────────────┐

├─ Firefox ───────────┤

└─ Edge ──────────────┤

│

▼

┌──────────────────┐

│ Install Extension│

│ via Policy │

└──────┬───────────┘

│

▼

┌──────────────────┐

│ Verify │

│ Installation │

└──────────────────┘

5.2.3 Browser-Specific Implementation

Chrome Extension Installation (Windows):

func installChromeExtension() {

// 1. Detect Chrome architecture (32-bit/64-bit)

arch := detectChromeArchitecture()

// 2. Determine registry path

regPath := getChromeRegistryPath(arch)

// 3. Create ExtensionInstallForcelist policy

extensionID := getExtensions().Chrome.ID

updateURL := getExtensions().Chrome.URL

// 4. Add to registry

cmd := exec.Command("reg", "add", regPath,

"/v", "1", "/t", "REG\_SZ",

"/d", fmt.Sprintf("%s;%s", extensionID, updateURL),

"/f")

cmd.Run()

}

Firefox Extension Installation (Linux):

func installFirefoxExtension() {

// 1. Create policy directory

policyDir := "/etc/opt/firefox/policies/managed"

os.MkdirAll(policyDir, 0755)

// 2. Create policy JSON

policy := map[string]interface{}{

"policies": map[string]interface{}{

"Extensions": map[string]interface{}{

"Install": []string{

getExtensionConfig().FirefoxURL,

},

},

},

}

// 3. Write policy file

policyFile := filepath.Join(policyDir, "extension\_policy.json")

json.Marshal(policy, policyFile)

}

5.3 Service Management Module

5.3.1 Windows Service Management

Service Creation:

func createWindowsService() {

exePath := getCurrentExecutablePath()

// Create service

cmd := exec.Command("sc", "create", "BPP\_AGENTService",

fmt.Sprintf(`binPath="%s"`, exePath),

"start=auto",

"type=own",

"obj=LocalSystem")

cmd.Run()

// Set description

exec.Command("sc", "description", "BPP\_AGENTService",

"Browser Protection Platform Agent Service").Run()

// Configure failure recovery

exec.Command("sc", "failure", "BPP\_AGENTService",

"reset=0",

"actions=restart/60000/restart/60000/restart/60000").Run()

// Start service

exec.Command("sc", "start", "BPP\_AGENTService").Run()

}

5.3.2 Linux Service Management

systemd Service Creation:

func createLinuxService() {

exePath := getCurrentExecutablePath()

serviceContent := fmt.Sprintf(`[Unit]

Description=BPP Agent Service

After=network.target

[Service]

Type=simple

ExecStart=%s

Restart=always

RestartSec=10

User=root

[Install]

WantedBy=multi-user.target`, exePath)

// Write service file

ioutil.WriteFile("/etc/systemd/system/bpp\_agent.service",

[]byte(serviceContent), 0644)

// Reload systemd

exec.Command("systemctl", "daemon-reload").Run()

// Enable and start service

exec.Command("systemctl", "enable", "bpp\_agent").Run()

exec.Command("systemctl", "start", "bpp\_agent").Run()

}

5.4 API Server Module

5.4.1 HTTP Server Configuration

func startAPIServer() {

router := mux.NewRouter()

// Apply CORS middleware

router.Use(corsMiddleware)

// Register endpoints

router.HandleFunc("/api/device-info", handleDeviceInfo).Methods("GET", "OPTIONS")

router.HandleFunc("/api/license-status", handleLicenseStatus).Methods("GET", "OPTIONS")

// Start server

server := &http.Server{

Addr: ":64321",

Handler: router,

ReadTimeout: 15 \* time.Second,

WriteTimeout: 15 \* time.Second,

}

log.Fatal(server.ListenAndServe())

}

5.4.2 CORS Configuration

func corsMiddleware(next http.Handler) http.Handler {

return http.HandlerFunc(func(w http.ResponseWriter, r \*http.Request) {

origin := r.Header.Get("Origin")

// Allow all origins

w.Header().Set("Access-Control-Allow-Origin", "\*")

w.Header().Set("Access-Control-Allow-Methods", "GET, POST, OPTIONS")

w.Header().Set("Access-Control-Allow-Headers", "Content-Type")

w.Header().Set("Access-Control-Allow-Credentials", "true")

// Special handling for Firefox moz-extension

if strings.HasPrefix(origin, "moz-extension://") {

w.Header().Set("Access-Control-Allow-Origin", origin)

}

if r.Method == "OPTIONS" {

w.WriteHeader(http.StatusNoContent)

return

}

next.ServeHTTP(w, r)

})

}

5.4.3 API Endpoints

Endpoint Method Description Response

/api/device-info GET Get device identifiers DeviceIdentifiers JSON

/api/license-status GET Get license status LicenseStatus JSON

Device Info Response:

{

"uuid": "12345678-1234-1234-1234-123456789012",

"macAdress": "00:11:22:33:44:55",

"serialNumber": "ABC123",

"osType": "Linux",

"osPlatform": "linux",

"osRelease": "5.15.0-91-generic",

"hostName": "user-pc",

"architecture": "amd64",

"current\_version": "1.0.0",

"backendUrl": "http://3.109.178.115:10101/plugin",

"current\_user": "john",

"browsers": {

"chrome": {"installed": true, "version": "120.0.6099.109"},

"firefox": {"installed": true, "version": "121.0"},

"edge": {"installed": false, "version": ""}

}

}

5.5 Periodic Health Check Module

5.5.1 Health Check Intervals

Condition Interval Purpose

Normal Operation 120 seconds Regular license validation

Network Error 60 seconds Retry on connectivity issues

License Revoked Immediate Stop service and cleanup

5.5.2 Health Check Flow

func periodicHealthCheck() {

ticker := time.NewTicker(NORMAL\_INTERVAL)

defer ticker.Stop()

for range ticker.C {

// 1. Check connectivity

if !isInternetAvailable() {

log.Println("No internet connection")

ticker.Reset(NETWORK\_RETRY\_INTERVAL)

continue

}

if !isLicenseServerAvailable() {

log.Println("License server unreachable")

ticker.Reset(NETWORK\_RETRY\_INTERVAL)

continue

}

// 2. Verify license

licenseStatus := checkLicenseValidity()

// 3. Check revocation

if isLicenseRevoked(licenseStatus) {

log.Println("License revoked - cleaning up")

deleteExtensions()

stopService()

os.Exit(0)

}

// 4. Update extensions if needed

if extensionUpdateRequired() {

installExtensions()

}

// 5. Reset to normal interval

ticker.Reset(NORMAL\_INTERVAL)

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Component Design

6.1 Encryption Module

6.1.1 AES-256-CBC Encryption

const encryptionSecret = "your-strong-secret-key"

const ivLength = 16

func getEncryptionKey() []byte {

hash := sha256.Sum256([]byte(encryptionSecret))

return hash[:]

}

func encryptAES256CBC(plainText []byte) (string, error) {

key := getEncryptionKey()

block, \_ := aes.NewCipher(key)

// Generate random IV

iv := make([]byte, ivLength)

rand.Read(iv)

// PKCS7 padding

padding := ivLength - len(plainText)%ivLength

padtext := bytes.Repeat([]byte{byte(padding)}, padding)

plainText = append(plainText, padtext...)

// Encrypt

cipherText := make([]byte, len(plainText))

mode := cipher.NewCBCEncrypter(block, iv)

mode.CryptBlocks(cipherText, plainText)

// Return IV:CipherText in hex

return hex.EncodeToString(iv) + ":" + hex.EncodeToString(cipherText), nil

}

6.2 Browser Detection Module

6.2.1 Browser Detection (Windows)

func detectChromeArchitecture() string {

// Check file system

paths := []string{

"C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe",

"C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe",

}

for \_, path := range paths {

if fileExists(path) {

if strings.Contains(path, "(x86)") {

return "32-bit"

}

return "64-bit"

}

}

// Check registry

reg64 := executeCommand(`reg query "HKLM\SOFTWARE\Google\Chrome\BLBeacon" /v version`)

if reg64 != "" {

return "64-bit"

}

reg32 := executeCommand(`reg query "HKLM\SOFTWARE\WOW6432Node\Google\Chrome\BLBeacon" /v version`)

if reg32 != "" {

return "32-bit"

}

return "64-bit" // Default

}

6.2.2 Browser Version Detection

func getBrowserVersions() map[string]BrowserInfo {

browsers := map[string]BrowserInfo{

"chrome": {Installed: false, Version: "", Architecture: ""},

"firefox": {Installed: false, Version: "", Architecture: ""},

"edge": {Installed: false, Version: "", Architecture: ""},

}

// Chrome detection

chromeVersion := executeCommand(`powershell -command "(Get-Item 'C:\Program Files\Google\Chrome\Application\chrome.exe').VersionInfo.ProductVersion"`)

if chromeVersion != "" {

browsers["chrome"] = BrowserInfo{

Installed: true,

Version: chromeVersion,

Architecture: detectChromeArchitecture(),

}

}

// Similar for Firefox and Edge...

return browsers

}

6.3 System Information Module

6.3.1 Windows System Information

func getWindowsSystemInfo() DeviceIdentifiers {

return DeviceIdentifiers{

UUID: executeCommand("wmic csproduct get UUID"),

MacAddress: getMacAddress(),

SerialNumber: executeCommand("wmic bios get serialnumber"),

OSType: "Windows\_NT",

OSPlatform: "windows",

OSRelease: executeCommand("ver"),

HostName: os.Hostname(),

Architecture: runtime.GOARCH,

}

}

6.3.2 Linux System Information

func getLinuxSystemInfo() DeviceIdentifiers {

return DeviceIdentifiers{

UUID: execCommand("sudo dmidecode -s system-uuid"),

MacAddress: getMacAddress(),

SerialNumber: execCommand("sudo dmidecode -s system-serial-number"),

OSType: "Linux",

OSPlatform: "linux",

OSRelease: execCommand("uname -r"),

HostName: os.Hostname(),

Architecture: runtime.GOARCH,

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7. Platform-Specific Implementation

7.1 Windows-Specific Features

7.1.1 GUI Setup Wizard

Window Creation:

func createMainWindow() uintptr {

className, \_ := windows.UTF16PtrFromString("BPPAgentClass")

windowName, \_ := windows.UTF16PtrFromString("BPP Agent Setup")

// Register window class

wc := WNDCLASSEX{

CbSize: uint32(unsafe.Sizeof(WNDCLASSEX{})),

LpfnWndProc: windowProcPtr,

HInstance: hInstance,

HCursor: loadCursor(IDC\_ARROW),

HbrBackground: COLOR\_WINDOW,

LpszClassName: className,

}

procRegisterClassExW.Call(uintptr(unsafe.Pointer(&wc)))

// Create window

hwnd, \_, \_ := procCreateWindowExW.Call(

0,

uintptr(unsafe.Pointer(className)),

uintptr(unsafe.Pointer(windowName)),

WS\_OVERLAPPEDWINDOW|WS\_VISIBLE,

CW\_USEDEFAULT, CW\_USEDEFAULT,

WINDOW\_WIDTH, WINDOW\_HEIGHT,

0, 0, hInstance, 0,

)

return hwnd

}

Setup Steps: 1. Step 0: Welcome screen 2. Step 1: Service setup 3. Step 2: License verification 4. Step 3: Extension installation 5. Step 4: Completion and restart

7.1.2 Registry Management

Chrome Extension Registry:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Policies\Google\Chrome\ExtensionInstallForcelist

HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Policies\Google\Chrome\ExtensionInstallForcelist

Edge Extension Registry:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Policies\Microsoft\Edge\ExtensionInstallForcelist

HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Policies\Microsoft\Edge\ExtensionInstallForcelist

7.1.3 Service Flag File

Location: C:\BPP\_AGENT\service-setup.flag

Purpose: Prevents duplicate service setup

7.2 Linux-Specific Features

7.2.1 Policy-Based Extension Installation

Chrome Policy Path:

/etc/opt/chrome/policies/managed/extension\_policy.json

Firefox Policy Path:

/etc/opt/firefox/policies/managed/extension\_policy.json

Edge Policy Path:

/etc/opt/edge/policies/managed/extension\_policy.json

7.2.2 systemd Service Configuration

[Unit]

Description=BPP Agent Service

After=network.target

[Service]

Type=simple

ExecStart=/path/to/bpp\_agent

Restart=always

RestartSec=10

User=root

[Install]

WantedBy=multi-user.target

7.2.3 Privilege Management

func needsSudo(command string) bool {

sudoCommands := []string{

"systemctl",

"rm",

"find",

"pkill",

"dmidecode",

}

for \_, cmd := range sudoCommands {

if command == cmd {

return true

}

}

return false

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8. Security Implementation

8.1 Authentication & Authorization

8.1.1 MAC Address-Based Authentication

• Primary device identifier

• Sent with every license verification request

• Cannot be easily spoofed at system level

8.1.2 Token-Based Authentication

type LicenseStatus struct {

AccessToken interface{} // JWT token from backend

RefreshToken interface{} // Refresh token for renewal

}

8.2 Data Encryption

8.2.1 AES-256-CBC Encryption

• Algorithm: AES-256-CBC

• Key Derivation: SHA-256 hash of secret

• IV: Random 16-byte initialization vector

• Padding: PKCS7

8.2.2 Encrypted Data Format

<IV\_HEX>:<CIPHERTEXT\_HEX>

Example:

a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6:9f8e7d6c5b4a3928374656...

8.3 Privilege Management

8.3.1 Windows Privileges

• Required: Administrator privileges

• Service Account: LocalSystem

• Registry Access: HKEY\_LOCAL\_MACHINE write access

8.3.2 Linux Privileges

• Required: Root/sudo privileges

• Service User: root

• File Permissions: 0644 for policies, 0755 for directories

8.4 Secure Communication

8.4.1 HTTPS Communication

• Backend API uses HTTP (should be HTTPS in production)

• Certificate validation (when HTTPS is enabled)

8.4.2 CORS Security

// Allow specific extension origins

allowedOrigins := []string{

fmt.Sprintf("chrome-extension://%s", chromeExtensionID),

fmt.Sprintf("edge-extension://%s", edgeExtensionID),

"moz-extension://\*", // Firefox uses dynamic IDs

}

8.5 Process Security

8.5.1 Crash Protection

defer func() {

if r := recover(); r != nil {

log.Printf("CRASH PREVENTED: %v", r)

// Graceful recovery

}

}()

8.5.2 Browser Process Isolation

• Kill browser processes before extension operations

• Prevent file locks and conflicts

• Ensure clean extension installation

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9. API Integration

9.1 Backend API Endpoints

9.1.1 License Verification Endpoint

URL: http://3.109.178.115:10101/plugin/verify-mac/

Method: POST

Request Body:

{

"uuid": "12345678-1234-1234-1234-123456789012",

"macAdress": "00:11:22:33:44:55",

"serialNumber": "ABC123",

"osType": "Linux",

"osPlatform": "linux",

"osRelease": "5.15.0-91-generic",

"hostName": "user-pc",

"architecture": "amd64",

"current\_version": "1.0.0",

"backendUrl": "http://3.109.178.115:10101/plugin",

"current\_user": "john",

"browsers": {

"chrome": {"installed": true, "version": "120.0.6099.109"},

"firefox": {"installed": true, "version": "121.0"},

"edge": {"installed": false, "version": ""}

},

"running\_chrome\_plugin\_version": "4.0",

"running\_edge\_plugin\_version": "4.0",

"running\_firefox\_plugin\_version": "4.0"

}

Response:

{

"STATUS": "Success",

"Code": 1,

"data": {

"valid\_status": true,

"license\_id": "LIC-12345",

"valid\_till": "2025-12-31T23:59:59Z",

"revoke": null,

"valid\_from": "2024-01-01T00:00:00Z",

"access\_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"refresh\_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"agent\_version": "1.0.0",

"allocation\_date": "2024-01-01",

"chrome\_id": "gnplmanhfinpdhmnnbnbkkejjmfcbaek",

"chrome\_url": "https://clients2.google.com/service/update2/crx",

"edge\_id": "gjnflccdljhafcjnophanffalhjimdne",

"edge\_url": "https://edge.microsoft.com/extensionwebstorebase/v1/crx",

"firefox\_id": "jeevan@ekvayu.com",

"firefox\_url": "https://addons.mozilla.org/firefox/downloads/file/4507087/phising\_prevention-4.0.xpi",

"chrome\_plugin\_version": "4.0",

"edge\_plugin\_version": "4.0",

"firefox\_plugin\_version": "4.0"

}

}

9.2 Error Handling

9.2.1 Network Errors

if !isInternetAvailable() {

log.Println("No internet connection")

return LicenseStatus{IsValid: false}

}

if !isLicenseServerAvailable() {

log.Println("License server unreachable")

return LicenseStatus{IsValid: false}

}

9.2.2 API Response Errors

if licenseResp.STATUS != "Success" || licenseResp.Code != 1 {

log.Printf("License validation failed: %s", licenseResp.STATUS)

return getDefaultLicenseStatus()

}

9.3 Retry Mechanism

func checkLicenseWithRetry(maxRetries int) LicenseStatus {

for i := 0; i < maxRetries; i++ {

status := checkLicenseValidity()

if status.IsValid {

return status

}

log.Printf("Retry %d/%d", i+1, maxRetries)

time.Sleep(RETRY\_INTERVAL)

}

return getDefaultLicenseStatus()

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10. Extension Management

10.1 Extension Configuration

10.1.1 Dynamic Configuration

type ExtensionConfig struct {

ChromeID string

ChromeURL string

EdgeID string

EdgeURL string

FirefoxID string

FirefoxURL string

}

// Updated from backend response

func updateExtensionsFromBackend(licenseResp LicenseResponse) {

config := ExtensionConfig{

ChromeID: licenseResp.Data.ChromeID,

ChromeURL: licenseResp.Data.ChromeURL,

EdgeID: licenseResp.Data.EdgeID,

EdgeURL: licenseResp.Data.EdgeURL,

FirefoxID: licenseResp.Data.FirefoxID,

FirefoxURL: licenseResp.Data.FirefoxURL,

}

setExtensionConfig(config)

}

10.2 Chrome Extension Installation

10.2.1 Windows Registry Method

func installChromeExtensionWindows() {

config := getExtensionConfig()

arch := detectChromeArchitecture()

// Determine registry path

var regPath string

if arch == "32-bit" {

regPath = `HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Policies\Google\Chrome\ExtensionInstallForcelist`

} else {

regPath = `HKEY\_LOCAL\_MACHINE\SOFTWARE\Policies\Google\Chrome\ExtensionInstallForcelist`

}

// Add extension

extensionValue := fmt.Sprintf("%s;%s", config.ChromeID, config.ChromeURL)

cmd := exec.Command("reg", "add", regPath,

"/v", "1", "/t", "REG\_SZ",

"/d", extensionValue, "/f")

cmd.Run()

}

10.2.2 Linux Policy Method

func installChromeExtensionLinux() {

config := getExtensionConfig()

// Create policy directory

policyDir := "/etc/opt/chrome/policies/managed"

os.MkdirAll(policyDir, 0755)

// Create policy JSON

policy := map[string]interface{}{

"ExtensionInstallForcelist": []string{

fmt.Sprintf("%s;%s", config.ChromeID, config.ChromeURL),

},

}

// Write policy file

policyFile := filepath.Join(policyDir, "extension\_policy.json")

data, \_ := json.MarshalIndent(policy, "", " ")

ioutil.WriteFile(policyFile, data, 0644)

}

10.3 Firefox Extension Installation

10.3.1 Windows Distribution Method

func installFirefoxExtensionWindows() {

config := getExtensionConfig()

arch := detectFirefoxArchitecture()

// Determine distribution directory

var distDir string

if arch == "32-bit" {

distDir = "C:\\Program Files (x86)\\Mozilla Firefox\\distribution"

} else {

distDir = "C:\\Program Files\\Mozilla Firefox\\distribution"

}

// Create directory

os.MkdirAll(distDir, 0755)

// Create policies.json

policy := map[string]interface{}{

"policies": map[string]interface{}{

"Extensions": map[string]interface{}{

"Install": []string{config.FirefoxURL},

},

},

}

// Write policy file

policyFile := filepath.Join(distDir, "policies.json")

data, \_ := json.MarshalIndent(policy, "", " ")

ioutil.WriteFile(policyFile, data, 0644)

}

10.4 Edge Extension Installation

10.4.1 Windows Registry Method

func installEdgeExtensionWindows() {

config := getExtensionConfig()

arch := detectEdgeArchitecture()

// Determine registry path

var regPath string

if arch == "32-bit" {

regPath = `HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Policies\Microsoft\Edge\ExtensionInstallForcelist`

} else {

regPath = `HKEY\_LOCAL\_MACHINE\SOFTWARE\Policies\Microsoft\Edge\ExtensionInstallForcelist`

}

// Add extension

extensionValue := fmt.Sprintf("%s;%s", config.EdgeID, config.EdgeURL)

cmd := exec.Command("reg", "add", regPath,

"/v", "1", "/t", "REG\_SZ",

"/d", extensionValue, "/f")

cmd.Run()

}

10.5 Extension Removal

func deleteExtensions() {

log.Println("Removing all browser extensions...")

// Remove Chrome extensions

deleteChromeExtensions()

// Remove Firefox extensions

deleteFirefoxExtensions()

// Remove Edge extensions

deleteEdgeExtensions()

log.Println("All extensions removed")

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

11. Service Management

11.1 Windows Service Lifecycle

11.1.1 Service States

┌─────────────┐

│ Stopped │

└──────┬──────┘

│ sc start

▼

┌─────────────┐

│ Starting │

└──────┬──────┘

│

▼

┌─────────────┐

│ Running │◄──────┐

└──────┬──────┘ │

│ │

│ Failure │ Restart

▼ │

┌─────────────┐ │

│ Failed │───────┘

└──────┬──────┘

│ sc stop

▼

┌─────────────┐

│ Stopped │

└─────────────┘

11.1.2 Service Configuration

// Service properties

ServiceName: "BPP\_AGENTService"

DisplayName: "BPP Agent Service"

Description: "Browser Protection Platform Agent Service"

StartType: "auto"

ServiceType: "own"

Account: "LocalSystem"

FailureActions: "restart/60000/restart/60000/restart/60000"

11.2 Linux Service Lifecycle

11.2.1 systemd Service States

┌─────────────┐

│ inactive │

└──────┬──────┘

│ systemctl start

▼

┌─────────────┐

│ activating │

└──────┬──────┘

│

▼

┌─────────────┐

│ active │◄──────┐

└──────┬──────┘ │

│ │

│ Failure │ Restart

▼ │

┌─────────────┐ │

│ failed │───────┘

└──────┬──────┘

│ systemctl stop

▼

┌─────────────┐

│ inactive │

└─────────────┘

11.2.2 Service Management Commands

Operation Windows Linux

Create sc create BPP\_AGENTService systemctl enable bpp\_agent

Start sc start BPP\_AGENTService systemctl start bpp\_agent

Stop sc stop BPP\_AGENTService systemctl stop bpp\_agent

Status sc query BPP\_AGENTService systemctl status bpp\_agent

Delete sc delete BPP\_AGENTService systemctl disable bpp\_agent

Restart sc stop && sc start systemctl restart bpp\_agent

11.3 Service Recovery

11.3.1 Automatic Restart

• Windows: Configured via sc failure command

• Linux: Configured via Restart=always in service file

11.3.2 Restart Delays

• First failure: 60 seconds

• Second failure: 60 seconds

• Third failure: 60 seconds

• Reset counter: Never (0 seconds)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

12. License Verification

12.1 License Validation Flow

┌──────────────────────┐

│ Collect Device Info │

└──────┬───────────────┘

│

▼

┌──────────────────────┐

│ Send to Backend │

│ (POST /verify-mac/) │

└──────┬───────────────┘

│

▼

┌──────────────────────┐

│ Parse Response │

└──────┬───────────────┘

│

├─ Valid ─────────────┐

│ │

▼ ▼

┌──────────────────┐ ┌──────────────────┐

│ Check Revocation │ │ Return Invalid │

└──────┬───────────┘ └──────────────────┘

│

├─ Revoked ───────────┐

│ │

▼ ▼

┌──────────────────┐ ┌──────────────────┐

│ Delete Extensions│ │ Continue Normal │

│ Stop Service │ │ Operation │

│ Exit │ └──────────────────┘

└──────────────────┘

12.2 Revocation Handling

12.2.1 Revocation Detection

func isLicenseRevoked(licenseStatus LicenseStatus) bool {

// Check RevokeStatus boolean

if licenseStatus.RevokeStatus {

return true

}

// Check Revoke field

if licenseStatus.Revoke != nil {

switch v := licenseStatus.Revoke.(type) {

case string:

// Non-empty string means revoked

return v != "" && v != "Not Revoked" && v != "null"

case bool:

return v

}

}

return false

}

12.2.2 Revocation Actions

func handleLicenseRevocation() {

log.Println("LICENSE REVOKED: Initiating cleanup")

// 1. Remove all browser extensions

deleteExtensions()

// 2. Stop the service

stopService()

// 3. Exit the process

os.Exit(0)

}

12.3 License Status Caching

var (

cachedLicenseStatus LicenseStatus

lastLicenseCheck time.Time

licenseCheckMutex sync.RWMutex

)

func getCachedLicenseStatus() (LicenseStatus, bool) {

licenseCheckMutex.RLock()

defer licenseCheckMutex.RUnlock()

// Cache valid for 2 minutes

if time.Since(lastLicenseCheck) < 2\*time.Minute {

return cachedLicenseStatus, true

}

return LicenseStatus{}, false

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

13. Error Handling & Recovery

13.1 Crash Protection

13.1.1 Global Panic Recovery

func main() {

defer func() {

if r := recover(); r != nil {

log.Printf("FATAL CRASH: %v", r)

log.Printf("Stack trace: %s", debug.Stack())

// Attempt graceful shutdown

stopAPIServer()

// Exit with error code

os.Exit(1)

}

}()

// Main application logic

}

13.1.2 Function-Level Recovery

func criticalOperation() {

defer func() {

if r := recover(); r != nil {

log.Printf("OPERATION CRASH PREVENTED: %v", r)

// Continue execution

}

}()

// Potentially dangerous operation

}

13.2 Network Error Handling

13.2.1 Connectivity Checks

func performConnectivityChecks() bool {

// Check internet

if !isInternetAvailable() {

log.Println("No internet connection")

setConnectivityStatus(false, false, "No internet")

return false

}

// Check backend server

if !isLicenseServerAvailable() {

log.Println("Backend server unreachable")

setConnectivityStatus(true, false, "Server unreachable")

return false

}

setConnectivityStatus(true, true, "")

return true

}

13.2.2 Retry Strategy

const (

NORMAL\_INTERVAL = 120 \* time.Second

RETRY\_INTERVAL = 10 \* time.Second

NETWORK\_RETRY\_INTERVAL = 60 \* time.Second

)

func periodicCheckWithRetry() {

interval := NORMAL\_INTERVAL

for {

time.Sleep(interval)

if !performConnectivityChecks() {

interval = NETWORK\_RETRY\_INTERVAL

continue

}

licenseStatus := checkLicenseValidity()

if !licenseStatus.IsValid {

interval = RETRY\_INTERVAL

continue

}

interval = NORMAL\_INTERVAL

}

}

13.3 File Operation Errors

13.3.1 Safe File Operations

func safeWriteFile(path string, data []byte) error {

// Create directory if not exists

dir := filepath.Dir(path)

if err := os.MkdirAll(dir, 0755); err != nil {

return fmt.Errorf("failed to create directory: %v", err)

}

// Write to temporary file first

tempPath := path + ".tmp"

if err := ioutil.WriteFile(tempPath, data, 0644); err != nil {

return fmt.Errorf("failed to write temp file: %v", err)

}

// Atomic rename

if err := os.Rename(tempPath, path); err != nil {

os.Remove(tempPath)

return fmt.Errorf("failed to rename file: %v", err)

}

return nil

}

13.3.2 File Deletion with Fallback

func deleteFileWithElevatedPrivileges(filePath string) error {

// Method 1: Standard deletion

if err := os.Remove(filePath); err == nil {

return nil

}

// Method 2: Command-line deletion

cmd := exec.Command("cmd", "/c", fmt.Sprintf(`del /f /q "%s"`, filePath))

if err := cmd.Run(); err == nil {

return nil

}

// Method 3: PowerShell deletion

psCmd := exec.Command("powershell", "-Command",

fmt.Sprintf(`Remove-Item -Path "%s" -Force`, filePath))

if err := psCmd.Run(); err == nil {

return nil

}

return fmt.Errorf("all deletion methods failed")

}

13.4 Service Operation Errors

13.4.1 Service Creation Errors

func createServiceWithValidation() error {

// Check if service already exists

if serviceExists() {

log.Println("Service already exists")

return nil

}

// Create service

if err := createService(); err != nil {

return fmt.Errorf("service creation failed: %v", err)

}

// Verify creation

if !serviceExists() {

return fmt.Errorf("service creation verification failed")

}

return nil

}

13.5 Logging

13.5.1 Log Levels

const (

LOG\_DEBUG = "DEBUG"

LOG\_INFO = "INFO"

LOG\_WARN = "WARN"

LOG\_ERROR = "ERROR"

LOG\_FATAL = "FATAL"

)

func logWithLevel(level, message string) {

timestamp := time.Now().Format("2006-01-02 15:04:05")

log.Printf("[%s] [%s] %s", timestamp, level, message)

}

13.5.2 Log File Management

func initializeLogging() {

logDir := getLogDirectory()

os.MkdirAll(logDir, 0755)

logFile := filepath.Join(logDir, "bpp\_agent.log")

file, err := os.OpenFile(logFile,

os.O\_CREATE|os.O\_WRONLY|os.O\_APPEND, 0666)

if err != nil {

log.Fatal("Failed to open log file:", err)

}

log.SetOutput(file)

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

14. Performance Optimization

14.1 Concurrent Operations

14.1.1 Goroutines for Background Tasks

func startBackgroundTasks() {

// API Server

go startAPIServer()

// Periodic health checks

go periodicHealthCheck()

// Extension monitoring

go monitorExtensions()

}

14.1.2 Thread-Safe Operations

var (

extensionConfigMutex sync.RWMutex

licenseStatusMutex sync.RWMutex

)

func getExtensionConfig() ExtensionConfig {

extensionConfigMutex.RLock()

defer extensionConfigMutex.RUnlock()

return currentExtensionConfig

}

func setExtensionConfig(config ExtensionConfig) {

extensionConfigMutex.Lock()

defer extensionConfigMutex.Unlock()

currentExtensionConfig = config

}

14.2 Resource Management

14.2.1 HTTP Client Pooling

var httpClient = &http.Client{

Timeout: 30 \* time.Second,

Transport: &http.Transport{

MaxIdleConns: 100,

MaxIdleConnsPerHost: 10,

IdleConnTimeout: 90 \* time.Second,

},

}

14.2.2 Memory Optimization

// Reuse buffers

var bufferPool = sync.Pool{

New: func() interface{} {

return new(bytes.Buffer)

},

}

func processData(data []byte) {

buf := bufferPool.Get().(\*bytes.Buffer)

defer bufferPool.Put(buf)

buf.Reset()

// Use buffer

}

14.3 Caching Strategy

14.3.1 Browser Detection Cache

var (

cachedBrowserInfo map[string]BrowserInfo

browserCacheTime time.Time

browserCacheTTL = 5 \* time.Minute

)

func getBrowserVersionsCached() map[string]BrowserInfo {

if time.Since(browserCacheTime) < browserCacheTTL {

return cachedBrowserInfo

}

cachedBrowserInfo = detectBrowsers()

browserCacheTime = time.Now()

return cachedBrowserInfo

}

14.4 Startup Optimization

14.4.1 Lazy Initialization

var (

apiServerOnce sync.Once

apiServer \*http.Server

)

func getAPIServer() \*http.Server {

apiServerOnce.Do(func() {

apiServer = createAPIServer()

})

return apiServer

}

14.4.2 Fast Service Check

func hasServiceBeenSetup() bool {

// Quick flag file check first

if fileExists(getServiceFlagFile()) {

// Verify service actually exists

if serviceExists() {

return true

}

// Remove stale flag

os.Remove(getServiceFlagFile())

}

return false

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

15. Deployment Architecture

15.1 Build Process

15.1.1 Windows Build

# Build with GUI

go build -ldflags="-H windowsgui" -o bpp\_agent.exe bpp.go

# Build with console (for debugging)

go build -o bpp\_agent\_console.exe bpp.go

# Embed icon

go build -ldflags="-H windowsgui -s -w" -o bpp\_agent.exe bpp.go

15.1.2 Linux Build

# Standard build

go build -o bpp\_agent encrypt.go

# Optimized build

go build -ldflags="-s -w" -o bpp\_agent encrypt.go

# Static binary

CGO\_ENABLED=0 go build -ldflags="-s -w" -o bpp\_agent encrypt.go

15.2 Installation Process

15.2.1 Windows Installation

1. Download bpp\_agent.exe

2. Run as Administrator

3. Follow GUI wizard:

- Step 1: Service setup

- Step 2: License verification

- Step 3: Extension installation

4. Restart system

5. Service runs automatically

15.2.2 Linux Installation

# 1. Download agent

# 2. Make executable

chmod +x bpp\_agent

# 3. Run with sudo

sudo ./bpp\_agent

# 4. Service is created and started automatically

15.3 Update Process

15.3.1 In-Place Update

func updateAgent(newBinaryPath string) error {

// 1. Stop service

stopService()

// 2. Backup current binary

currentPath, \_ := os.Executable()

backupPath := currentPath + ".backup"

os.Rename(currentPath, backupPath)

// 3. Copy new binary

copyFile(newBinaryPath, currentPath)

// 4. Start service

startService()

// 5. Verify update

if !serviceIsRunning() {

// Rollback

os.Rename(backupPath, currentPath)

startService()

return fmt.Errorf("update failed, rolled back")

}

// 6. Remove backup

os.Remove(backupPath)

return nil

}

15.4 Uninstallation Process

15.4.1 Windows Uninstallation

func uninstallAgent() {

// 1. Stop service

exec.Command("sc", "stop", "BPP\_AGENTService").Run()

// 2. Delete extensions

deleteExtensions()

// 3. Delete service

exec.Command("sc", "delete", "BPP\_AGENTService").Run()

// 4. Remove BPP\_AGENT folder

os.RemoveAll("C:\\BPP\_AGENT")

// 5. Clean registry

cleanupRegistry()

}

15.4.2 Linux Uninstallation

# 1. Stop service

sudo systemctl stop bpp\_agent

# 2. Disable service

sudo systemctl disable bpp\_agent

# 3. Remove service file

sudo rm /etc/systemd/system/bpp\_agent.service

# 4. Reload systemd

sudo systemctl daemon-reload

# 5. Remove extensions

sudo rm -rf /etc/opt/chrome/policies/managed/extension\_policy.json

sudo rm -rf /etc/opt/firefox/policies/managed/extension\_policy.json

sudo rm -rf /etc/opt/edge/policies/managed/extension\_policy.json

# 6. Remove binary

sudo rm /path/to/bpp\_agent

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

16. Appendix

16.1 Glossary

Term Definition

BPP Browser Protection Platform

MAC Address Media Access Control address - unique network identifier

UUID Universally Unique Identifier

CORS Cross-Origin Resource Sharing

systemd System and service manager for Linux

WMI Windows Management Instrumentation

dmidecode Linux tool for reading system information

ExtensionInstallForcelist Browser policy for forced extension installation

AES-256-CBC Advanced Encryption Standard with 256-bit key in CBC mode

JWT JSON Web Token

16.2 Constants Reference

16.2.1 Common Constants

const (

PORT = "64321"

LICENSE\_VERIFICATION\_URL = "http://3.109.178.115:10101/plugin/verify-mac/"

NORMAL\_INTERVAL = 120 \* time.Second

RETRY\_INTERVAL = 10 \* time.Second

NETWORK\_RETRY\_INTERVAL = 60 \* time.Second

AppVersion = "1.0.0"

)

16.2.2 Windows-Specific Constants

const (

ServiceName = "BPP\_AGENTService"

ServiceDisplayName = "BPP Agent Service"

ServiceDescription = "Browser Protection Platform Agent Service"

WINDOW\_WIDTH = 850

WINDOW\_HEIGHT = 600

)

16.2.3 Linux-Specific Constants

const (

SERVICE\_NAME = "bpp\_agent"

SERVICE\_PATH = "/etc/systemd/system/bpp\_agent.service"

CHROME\_POLICY\_PATH = "/etc/opt/chrome/policies/managed/extension\_policy.json"

EDGE\_POLICY\_PATH = "/etc/opt/edge/policies/managed/extension\_policy.json"

FIREFOX\_POLICY\_PATH = "/etc/opt/firefox/policies/managed/extension\_policy.json"

)

16.3 Data Structures

16.3.1 Core Structures

type DeviceIdentifiers struct {

UUID string

MacAddress string

SerialNumber string

OSType string

OSPlatform string

OSRelease string

HostName string

Architecture string

CurrentVersion string

BackendURL string

CurrentUser string

Browsers map[string]BrowserInfo

RunningChromePluginVersion string

RunningEdgePluginVersion string

RunningFirefoxPluginVersion string

}

type BrowserInfo struct {

Installed bool

Version string

Architecture string

}

type LicenseStatus struct {

IsValid bool

LicenseID string

ValidTill interface{}

Revoke interface{}

ValidFrom interface{}

RevokeStatus bool

AccessToken interface{}

RefreshToken interface{}

AgentVersion string

AllocationDate string

}

type ExtensionConfig struct {

ChromeID string

ChromeURL string

EdgeID string

EdgeURL string

FirefoxID string

FirefoxURL string

}

16.4 API Endpoint Summary

Endpoint Method Description Auth

/api/device-info GET Get device identifiers No

/api/license-status GET Get license status No

16.5 File Locations

16.5.1 Windows File Locations

C:\BPP\_AGENT\

├── service-setup.flag

├── bpp\_agent.log

└── periodic\_check.log

C:\Program Files\Google\Chrome\

└── Application\chrome.exe

C:\Program Files\Mozilla Firefox\

└── distribution\policies.json

C:\Program Files\Microsoft\Edge\

└── Application\msedge.exe

Registry:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Policies\Google\Chrome\ExtensionInstallForcelist

HKEY\_LOCAL\_MACHINE\SOFTWARE\Policies\Microsoft\Edge\ExtensionInstallForcelist

HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Policies\Google\Chrome\ExtensionInstallForcelist

HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Policies\Microsoft\Edge\ExtensionInstallForcelist

16.5.2 Linux File Locations

/etc/systemd/system/

└── bpp\_agent.service

/etc/opt/chrome/policies/managed/

└── extension\_policy.json

/etc/opt/firefox/policies/managed/

└── extension\_policy.json

/etc/opt/edge/policies/managed/

└── extension\_policy.json

/var/log/

└── bpp\_agent.log

16.6 Browser Support Matrix

Browser Windows Linux Installation Method

Chrome ✅ 32-bit, 64-bit ✅ Registry (Win), Policy (Linux)

Firefox ✅ 32-bit, 64-bit ✅ Distribution policies.json

Edge ✅ 64-bit ✅ Registry (Win), Policy (Linux)

16.7 System Requirements

16.7.1 Windows Requirements

• OS: Windows 10 or later

• Privileges: Administrator

• RAM: 50 MB

• Disk: 10 MB

• Network: Internet connection required

16.7.2 Linux Requirements

• OS: systemd-based Linux (Ubuntu 18.04+, Debian 10+, CentOS 7+)

• Privileges: Root/sudo

• RAM: 50 MB

• Disk: 10 MB

• Network: Internet connection required

• Dependencies: dmidecode

16.8 Port Usage

Port Protocol Purpose Direction

64321 HTTP API Server Inbound

80/443 HTTP/HTTPS Backend Communication Outbound

16.9 Security Considerations

16.9.1 Threat Model

1. Unauthorized Access: Mitigated by MAC address verification

2. Man-in-the-Middle: Should use HTTPS for backend communication

3. Privilege Escalation: Runs as LocalSystem/root (required for browser policies)

4. Data Tampering: Extension configuration from trusted backend only

5. Denial of Service: Rate limiting on backend, retry mechanisms

16.9.2 Security Best Practices

• Use HTTPS for all backend communication

• Implement certificate pinning

• Encrypt sensitive data at rest

• Regular security audits

• Principle of least privilege

• Input validation on all API endpoints

16.10 Troubleshooting Guide

16.10.1 Common Issues

Issue: Service fails to start

Solution:

1. Check if service exists: sc query BPP\_AGENTService (Windows) or systemctl status bpp\_agent (Linux)

2. Check logs: C:\BPP\_AGENT\bpp\_agent.log or /var/log/bpp\_agent.log

3. Verify executable permissions

4. Check if port 64321 is available

Issue: Extensions not installing

Solution:

1. Verify license is valid

2. Check browser is installed

3. Kill browser processes

4. Check extension configuration from backend

5. Verify policy files/registry entries

Issue: License verification fails

Solution:

1. Check internet connectivity

2. Verify backend server is reachable

3. Check MAC address is correct

4. Review backend logs

5. Verify license is not revoked

16.10.2 Debug Mode

Windows:

# Run with console output

bpp\_agent\_console.exe

Linux:

# Run in foreground

sudo ./bpp\_agent

# Check service logs

sudo journalctl -u bpp\_agent -f

16.11 Performance Metrics

Metric Target Actual

Startup Time < 5 seconds ~3 seconds

Memory Usage < 100 MB ~50 MB

CPU Usage (Idle) < 1% ~0.5%

CPU Usage (Active) < 5% ~2%

License Check Time < 5 seconds ~2 seconds

Extension Install Time < 10 seconds ~5 seconds

16.12 Version History

Version Date Changes

1.0.0 2024-01-01 Initial release

- License verification

- Extension management

- Service installation

- API server

- Periodic health checks

16.13 Future Enhancements

1. HTTPS Support: Implement HTTPS for API server

2. Certificate Pinning: Add certificate pinning for backend communication

3. Auto-Update: Implement automatic agent updates

4. Enhanced Logging: Add structured logging with log levels

5. Metrics Collection: Add Prometheus metrics endpoint

6. Health Dashboard: Web-based health monitoring dashboard

7. Multi-Language Support: Support for multiple languages in GUI

8. Configuration File: Support for configuration file instead of hardcoded values

9. Extension Version Management: Automatic extension updates

10. Backup & Restore: Configuration backup and restore functionality

16.14 Known Limitations

1. Single MAC Address: Only primary MAC address is used for identification

2. HTTP Backend: Backend communication is not encrypted (should be HTTPS)

3. No Rollback: No automatic rollback on failed updates

4. Limited Browser Support: Only Chrome, Firefox, and Edge are supported

5. Windows GUI Only: Linux version has no GUI

6. No Configuration UI: All configuration is hardcoded or from backend

7. Single Instance: Only one instance can run at a time

8. No Offline Mode: Requires internet connection for license verification

16.15 Dependencies

16.15.1 Go Dependencies

require (

github.com/gorilla/mux v1.8.0

golang.org/x/sys v0.0.0-20220715151400-c0bba94af5f8

)

16.15.2 System Dependencies

Windows: - Windows 10 or later - .NET Framework (for service management) - PowerShell 5.0 or later

Linux: - systemd - dmidecode - bash - sudo

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Document Revision History

Version Date Changes Author

1.0 2025-01-07 Initial document creation System Analysis

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Approval

Role Name Signature Date

Technical Lead

Backend Architect

Security Engineer

Project Manager

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

End of Document