

Forum:[Hack Forums](#)

Thread: [High-performance, lightweight Remote Administration Tool \(C++ Native, No CRT, No STL\)](#)

Xinfo - May 16, 2026, 10:20 am

Hey everyone,

I've recently finished developing a Remote Administration Tool (SokaRemote) fully coded in native C++ with zero CRT and zero STL dependencies.

A quick breakdown of the project:

Extremely Lightweight: The client stub size sits right around 24KB.

Highly Scalable: Thanks to a custom-built DLL loader, you can stack hundreds of additional features/modules without increasing that 24KB baseline stub size.

Rock-Solid Stability: I've recently pushed a major stability update. It runs flawlessly without any crashes or unexpected drops.

Before I look into setting up a sales thread or officially promoting it here, I wanted to get the community's feedback. Do you think there's a solid market for a custom, low-level build like this on HF? Would anyone be interested in purchasing or testing it? I'd love to know if you think it's worth the value.

Thanks for reading, and appreciate your insights!

I don't mean to sound dismissive of other tools on the market, as they certainly have their own audience and purposes. However, my personal development philosophy focuses more on core efficiency rather than flashy features. A lot of public tools rely on high-level frameworks like C#, which are great for rapid development and heavy UI, but they naturally introduce more bloat and dependency.

For this project, I really wanted to challenge myself to build something strictly minimal and stable. That's why I chose a native C++ approach with zero dependencies—to prioritize performance and stealth over unnecessary overhead.

Since a few members expressed curiosity about the current state of my project, SokaRemote Professional v3.0, I am writing this follow-up post to share the latest updates on its development.

Currently, the client-side stub architecture is completely finished. I am now focusing entirely on building and polishing the controller/server side using Delphi. My main objective at this stage is squashing minor bugs, optimizing network routines, and maximizing overall stability.

Someone previously requested some screenshots, so I would love to showcase the current interface and some core modules below.

This is the main panel where active connections are managed. When you right-click on a target, a

comprehensive context menu appears with a wide range of features.

A major architectural highlight of SokaRemote is its Custom Dynamic Loading (Zero-Import Stub) engine. Thanks to this low-level design, even if I add hundreds of additional features to the framework, the client stub size remains rigidly locked at around 28 KB. It does not bloat because modules are dynamically structured and streamed. The server is designed to handle this seamlessly, making the agent extremely lightweight and stealthy.

2. Screen Capture & Remote Control Module

This is the Remote Desktop form. It stream-decodes incoming frame packets smoothly on a dedicated background thread to prevent UI freezing.

Real-time Status Monitoring: The status bar provides absolute transparency with dedicated panels showing current streaming status, Resolution (Res), compressed Size (KB), and real-time frame rates (FPS).

Interactive Control: It supports smooth keyboard capturing, mouse movement, click synchronization, and custom JPEG quality/monitor selection.

3. Advanced Process Manager (Tree Structure)

This module maps the target's running tasks. Instead of a messy flat list, it utilizes a clean 2-pass tree structure algorithm to clearly separate Parent and Child processes.

Granular Details: Displays precise Process Name, PID, Architecture (Arch), and full executable Path.

Control Actions: Features a built-in search filter and allows administrators to Terminate (Kill), Suspend (Pause), or Resume processes on the fly.

4. Interactive File Manager & Transfers

The File Manager is split into two major operational tabs: File Explorer and File Transfers.

File Explorer: Allows full multi-drive navigation with real-time Windows system icon mapping. It fully supports standard operational hotkeys (F5 for Refresh, Del for Delete, F2 for Rename, Ctrl+C/X/V, etc.).

File Transfers: Houses asynchronous multi-threaded uploading and downloading tasks. It displays transfer progress with custom-drawn green progress bars, real-time metrics, transfer volume, and calculated speeds. It also includes an option to "Browse Local Downloads" to open the local storage directory immediately via context menu.

5. Startup Manager

This form interacts directly with the target's registry run keys to manage persistence.

Categorized Views: It fetches and parses entries into designated scopes: CurrentUser\Run, CurrentUser\RunOnce, LocalMachine\Run, and LocalMachine\RunOnce.

Registry Management: Provides an intuitive GUI dialog to seamlessly Add new startup items, Edit

existing values/paths, or completely Remove persistence entries.

Current Status

Overall, the core architecture is performing incredibly well and working exactly as intended. I am currently focusing on optimizing the server's memory management, tuning the socket handling, and resolving minor visual layout bugs.

I will keep updating this thread as development progresses toward a stable release. Feel free to leave any technical questions or feedback below!

Best regards.

Update: Private Beta Program (Limited Slots)

The development of SokaRemote Professional v3.0 has reached its final stage. The core architecture is rock-solid, module streaming is seamless, and the server-side memory management has been fully optimized for high-traffic environments.

To ensure peak performance and compatibility before the public release, I am opening a limited number of spots for the Private Beta Program.

Why the \$600 USD participation fee?

Building a tool with Zero CRT and Zero STL dependencies in Native C++ is a massive undertaking that trades developer convenience for absolute efficiency. While high-level frameworks rely on bloated dependencies, SokaRemote is forged from the metal up.

Think of it this way: most tools on the market are like heavy, feature-laden SUVs—they work, but they are loud, slow to turn, and leave a huge footprint. SokaRemote is like a carbon-fiber supercar. With a stub size locked under 30KB, it offers superior stealth, unmatched connection speed, and a level of stability that simply cannot be replicated by high-level wrappers. You are paying for the architectural precision, the lightweight footprint, and the uncompromising performance that defines this project.

What you will get:

Early Access: Immediate, exclusive access to the current build of SokaRemote Pro v3.0.

Priority Support: Direct 1-on-1 communication with me for bug reports and custom performance tuning.

2-Month Update Cycle: Full access to all hotfixes and feature expansions for the next 2 months. After this period, your build remains fully functional, but it will no longer receive automated updates.

Beta Participation Fee: \$600 USD

If you are a serious professional who values low-level stability, stealth, and elite-tier engineering, please feel free to send me a private message (PM) for more details on how to join.

Note: This is an exclusive opportunity for serious testers. My priority for this release is quality and stability over mass accessibility.

Best regards.

Looking for beta testers to help iron out bugs and test features. If you are interested in helping post here and I will DM you if accepted.

0101010111100010000 - May 16, 2026, 10:52 am

(May 16, 2026 10:20 AM)suka001xx Wrote: [\[CITE\]](#) Hey everyone,

I've recently finished developing a Remote Administration Tool (RAT) fully coded in native C++ with zero CRT and zero STL dependencies.

A quick breakdown of the project:

Extremely Lightweight: The client stub size sits right around 24KB.

Highly Scalable: Thanks to a custom-built DLL loader, you can stack hundreds of additional features/modules without increasing that 24KB baseline stub size.

Rock-Solid Stability: I've recently pushed a major stability update. It runs flawlessly without any crashes or unexpected drops.

Before I look into setting up a sales thread or officially promoting it here, I wanted to get the community's feedback. Do you think there's a solid market for a custom, low-level build like this on HF? Would anyone be interested in purchasing or testing it? I'd love to know if you think it's worth the value.

Thanks for reading, and appreciate your insights!

I don't mean to sound dismissive of other tools on the market, as they certainly have their own audience and purposes. However, my personal development philosophy focuses more on core efficiency rather than flashy features. A lot of public tools rely on high-level frameworks like C#, which are great for rapid development and heavy UI, but they naturally introduce more bloat and dependency.

For this project, I really wanted to challenge myself to build something strictly minimal and stable. That's why I chose a native C++ approach with zero dependencies—to prioritize performance and stealth over unnecessary overhead.

sounds good, usually stealth and long-term stability is much more beneficial than loud bloat. Care to share more about the features?

Xinfo - May 16, 2026, 11:01 am

(May 16, 2026 10:52 AM)0101010111100010000 Wrote: [\[CITE\]](#) sounds good, usually stealth and long-term stability is much more beneficial than loud bloat. Care to share more about the features?

Thanks for your insightful comment! You completely understand the core philosophy behind this project. I firmly believe that a minimal runtime footprint and long-term stability are far more valuable than a bloated, heavy client.

To answer your question about the current features and how they are structured:

The entire client is built using pure Win32 API with completely zero CRT/STL dependencies (/NODEFAULTLIB), which allows it to maintain that 24KB size. To keep the Import Address Table (IAT) completely clean and minimal under the hood, I've also implemented API Hashing to resolve all necessary system functions dynamically at runtime.

Currently, the framework has a solid baseline of essential management capabilities implemented, including:

Monitoring & Control: High-performance Screen Stream, Webcam Stream, and a fully functional Hidden VNC.

System Management: Low-level Process Manager, File Manager, Startup Manager, Service Manager, Window Manager, and Registry Editor.

Execution & Utilities: Remote CMD, Remote Scripting, Clipboard Manager, Installed Programs list, and an Offline Keylogger.

The biggest advantage of this design is its scalability. These features are just the beginning; because of the custom memory-mapping architecture, any new or customized capabilities can be seamlessly added later as plug-in modules without changing or increasing the 24KB core stub footprint at all.

I'm currently finalizing some minor code optimizations and structural polishes. I'd love to hear if there are any specific modules or features you think are absolute essentials for a low-level build like this!

0101010111100010000 - May 16, 2026, 11:30 am

(May 16, 2026 11:01 AM)suka001xx Wrote: [\[CITE\]](#) Thanks for your insightful comment! You completely understand the core philosophy behind this project. I firmly believe that a minimal runtime footprint and long-term stability are far more valuable than a bloated, heavy client.

To answer your question about the current features and how they are structured:

The entire client is built using pure Win32 API with completely zero CRT/STL dependencies (/NODEFAULTLIB), which allows it to maintain that 24KB size. To keep the Import Address Table (IAT) completely clean and minimal under the hood, I've also implemented API Hashing to resolve all necessary system functions dynamically at runtime.

Currently, the framework has a solid baseline of essential management capabilities implemented, including:

Monitoring & Control: High-performance Screen Stream, Webcam Stream, and a fully functional Hidden VNC.

System Management: Low-level Process Manager, File Manager, Startup Manager, Service Manager, Window Manager, and Registry Editor.

Execution & Utilities: Remote CMD, Remote Scripting, Clipboard Manager, Installed Programs list, and an Offline Keylogger.

The biggest advantage of this design is its scalability. These features are just the beginning; because of the custom memory-mapping architecture, any new or customized capabilities can be seamlessly added later as plug-in modules without changing or increasing the 24KB core stub footprint at all.

I'm currently finalizing some minor code optimizations and structural polishes. I'd love to hear if there are any specific modules or features you think are absolute essentials for a low-level build like this!

That certainly sounds like an epic project, congratulations man. And I think considering the file size and robustness of the script, the current features should definitely do the job. Would you mind sharing your ideas for the crypting of the script and what technique it would be using to hide itself(process hollowing, dll injection etc.?) Just curious to see what works in 2026. Cheers man

OffCode - May 17, 2026, 1:36 am

Share some pictures

Xinfo - May 17, 2026, 7:28 am

(May 17, 2026 01:36 AM)M A G U Wrote: [\[CITE\]](#) Share some pictures

Thanks for the interest! I've just updated the main post with the screenshots you requested.

Xinfo - May 17, 2026, 11:38 am

(May 17, 2026 11:34 AM)4chan.org/b/ Wrote: [\[CITE\]](#) Im creating my own to release for the market, had a quick glimpse over this. Looks promising doesnt look like slop AI. The remote desktop UI looks a lot like one thats already in the market (the layout) other than that, congrats on it being native.

Will this be free?

Thank you for the compliments and the solid feedback! You have a very sharp eye.

To be honest, I am not much of a designer, so I occasionally look at well-established tools in the market for layout and UI inspiration. I really appreciate you noticing the native architecture, as keeping the stub lightweight and optimized was a major focus for me.

As for your question regarding whether it will be free, I am currently planning to offer it under a subscription-based model once it officially hits the market.

Best of luck with your own project as well, I look forward to seeing your release on the market!

dench2025 - May 18, 2026, 4:21 am

I hope it opens soon

d3athgiver - May 18, 2026, 5:19 am

(May 17, 2026 11:38 AM)suka001xx Wrote: [\[CITE\]](#) Thank you for the compliments and the solid feedback! You have a very sharp eye.

To be honest, I am not much of a designer, so I occasionally look at well-established tools in the market for layout and UI inspiration. I really appreciate you noticing the native architecture, as keeping the stub lightweight and optimized was a major focus for me.

As for your question regarding whether it will be free, I am currently planning to offer it under a subscription-based model once it officially hits the market.

Best of luck with your own project as well, I look forward to seeing your release on the market!

You uave any plan to offer a version ubcrypted and one with cryoting for a higher price?? My friend is at same Stade of you but on his cryoter and I know he was hoping to have more feedback and less sandbox test maybe you guys can help each other and be both winners.. I know he have sine silent exploit fud working but I confirm you it don't have any builder thi gs done yet.

(May 17, 2026 11:38 AM)suka001xx Wrote: [\[CITE\]](#) Thank you for the compliments and the solid feedback! You have a very sharp eye.

To be honest, I am not much of a designer, so I occasionally look at well-established tools in the market for layout and UI inspiration. I really appreciate you noticing the native architecture, as keeping the stub lightweight and optimized was a major focus for me.

As for your question regarding whether it will be free, I am currently planning to offer it under a subscription-based model once it officially hits the market.

Best of luck with your own project as well, I look forward to seeing your release on the market!

Xinfo - May 18, 2026, 8:22 am

(May 18, 2026 05:19 AM)d3athgiver Wrote: [\[CITE\]](#) You uave any plan to offer a version ubcrypted and one with cryoting for a higher price?? My friend is at same Stade of you but on his cryoter and I know he was hoping to have more feedback and less sandbox test maybe you guys can help each other and be both winners.. I know he have sine silent exploit fud working but I confirm you it don't have any builder thi gs done yet.

Thank you for the interesting offer and input.

However, I am currently entirely focused on debugging my own Delphi controller and optimizing the core framework based on my strictly planned roadmap. I don't have any plans for external integration or partnerships at this stage of development.

Soka Professional will be fully ready as a complete, independent package once it officially hits the market under our subscription model. You and your friend are more than welcome to experience its full capability when the official premium version is released soon.

(May 18, 2026 04:21 AM)dench2025 Wrote: [\[CITE\]](#) I hope it opens soon
Thank you for the support!

I am doing my best to polish the server and fix the remaining bugs as quickly as possible. Stay tuned, the official release is just around the corner!

jiawqt - May 18, 2026, 9:49 am

good project

CL1B - May 18, 2026, 9:59 am

I kind of like this, good luck with dev!

Xinfo - May 18, 2026, 10:04 am

(May 18, 2026 09:59 AM)MarlboroMan Wrote: [\[CITE\]](#) I kind of like this, good luck with dev!
Thank you so much for the support! I really appreciate the encouragement from an experienced member like you.

I'm working hard on the final polish and hope to bring a solid release to the market soon. Thanks again!

klop1100 - May 19, 2026, 10:23 am

This looks amazing! The design is super clean, and the features seem incredibly well thought out. Fingers crossed the official release is coming soon because I can't wait to try it!

zcoderrx - May 19, 2026, 7:05 pm

It seems quite different from other poorly-made, AI-generated RATs being sold around this forum. The level of completion looks highly impressive, and it definitely doesn't look like a low-effort AI

product. If this is truly built on native C++ with no CRT and no STL, I honestly think it's the best tool available here. I really hope it gets released soon!

coocochi - May 19, 2026, 7:41 pm

This looks solid, would like to see all the features once its released. GL OP

OtterLover - May 20, 2026, 2:34 am

Try to make it quickly and fully. I want to try it once I make it. The UI is familiar.

Xinfo - May 23, 2026, 1:06 pm

[quote="soodalpie" pid='62958004' dateline='1779269648']

Try to make it quickly and fully. I want to try it once I make it. The UI is familiar.

[/quote]

? [warn]

[quote="suka001xx" pid='62956238' dateline='1778952009']

Hey everyone,

I've recently finished developing a Remote Administration Tool (SokaRemote) fully coded in native C++ with zero CRT and zero STL dependencies.

A quick breakdown of the project:

Extremely Lightweight: The client stub size sits right around 24KB.

Highly Scalable: Thanks to a custom-built DLL loader, you can stack hundreds of additional features/modules without increasing that 24KB baseline stub size.

Rock-Solid Stability: I've recently pushed a major stability update. It runs flawlessly without any crashes or unexpected drops.

Before I look into setting up a sales thread or officially promoting it here, I wanted to get the community's feedback. Do you think there's a solid market for a custom, low-level build like this on HF? Would anyone be interested in purchasing or testing it? I'd love to know if you think it's worth the value.

Thanks for reading, and appreciate your insights!

I don't mean to sound dismissive of other tools on the market, as they certainly have their own audience and purposes. However, my personal development philosophy focuses more on core efficiency rather than flashy features. A lot of public tools rely on high-level frameworks like C#, which are great for rapid development and heavy UI, but they naturally introduce more bloat and dependency.

For this project, I really wanted to challenge myself to build something strictly minimal and stable. That's why I chose a native C++ approach with zero dependencies—to prioritize performance and stealth over unnecessary overhead.

Since a few members expressed curiosity about the current state of my project, SokaRemote Professional v3.0, I am writing this follow-up post to share the latest updates on its development.

Currently, the client-side stub architecture is completely finished. I am now focusing entirely on

building and polishing the controller/server side using Delphi. My main objective at this stage is squashing minor bugs, optimizing network routines, and maximizing overall stability.

Someone previously requested some screenshots, so I would love to showcase the current interface and some core modules below.

This is the main panel where active connections are managed. When you right-click on a target, a comprehensive context menu appears with a wide range of features.

A major architectural highlight of SokaRemote is its Custom Dynamic Loading (Zero-Import Stub) engine. Thanks to this low-level design, even if I add hundreds of additional features to the framework, the client stub size remains rigidly locked at around 28 KB. It does not bloat because modules are dynamically structured and streamed. The server is designed to handle this seamlessly, making the agent extremely lightweight and stealthy.

2. Screen Capture & Remote Control Module

This is the Remote Desktop form. It stream-decodes incoming frame packets smoothly on a dedicated background thread to prevent UI freezing.

Real-time Status Monitoring: The status bar provides absolute transparency with dedicated panels showing current streaming status, Resolution (Res), compressed Size (KB), and real-time frame rates (FPS).

Interactive Control: It supports smooth keyboard capturing, mouse movement, click synchronization, and custom JPEG quality/monitor selection.

3. Advanced Process Manager (Tree Structure)

This module maps the target's running tasks. Instead of a messy flat list, it utilizes a clean 2-pass tree structure algorithm to clearly separate Parent and Child processes.

Granular Details: Displays precise Process Name, PID, Architecture (Arch), and full executable Path.

Control Actions: Features a built-in search filter and allows administrators to Terminate (Kill), Suspend (Pause), or Resume processes on the fly.

4. Interactive File Manager & Transfers

The File Manager is split into two major operational tabs: File Explorer and File Transfers.

File Explorer: Allows full multi-drive navigation with real-time Windows system icon mapping. It fully supports standard operational hotkeys (F5 for Refresh, Del for Delete, F2 for Rename, Ctrl+C/X/V, etc.).

File Transfers: Houses asynchronous multi-threaded uploading and downloading tasks. It displays transfer progress with custom-drawn green progress bars, real-time metrics, transfer volume, and calculated speeds. It also includes an option to "Browse Local Downloads" to open the local storage directory immediately via context menu.

5. Startup Manager

This form interacts directly with the target's registry run keys to manage persistence.

Categorized Views: It fetches and parses entries into designated scopes: `CurrentUser\Run`, `CurrentUser\RunOnce`, `LocalMachine\Run`, and `LocalMachine\RunOnce`.

Registry Management: Provides an intuitive GUI dialog to seamlessly Add new startup items, Edit existing values/paths, or completely Remove persistence entries.

Current Status

Overall, the core architecture is performing incredibly well and working exactly as intended. I am currently focusing on optimizing the server's memory management, tuning the socket handling, and resolving minor visual layout bugs.

I will keep updating this thread as development progresses toward a stable release. Feel free to leave any technical questions or feedback below!

Best regards.

CyberneticsX - June 15, 2026, 12:07 am

Looks good if not another AI Slop. When it's getting released?

KevenLionardo - June 22, 2026, 3:55 pm

here this is real RAT <=

x64assembler - August 27, 2026, 2:04 am

(May 16, 2026 10:20 AM)suka001xx Wrote: [\[CITE\]](#) Hey everyone,

I've recently finished developing a Remote Administration Tool (SokaRemote) fully coded in native C++ with zero CRT and zero STL dependencies.

A quick breakdown of the project:

Extremely Lightweight: The client stub size sits right around 24KB.

Highly Scalable: Thanks to a custom-built DLL loader, you can stack hundreds of additional features/modules without increasing that 24KB baseline stub size.

Rock-Solid Stability: I've recently pushed a major stability update. It runs flawlessly without any crashes or unexpected drops.

Before I look into setting up a sales thread or officially promoting it here, I wanted to get the community's feedback. Do you think there's a solid market for a custom, low-level build like this on HF? Would anyone be interested in purchasing or testing it? I'd love to know if you think it's worth the value.

Thanks for reading, and appreciate your insights!

I don't mean to sound dismissive of other tools on the market, as they certainly have their own audience and purposes. However, my personal development philosophy focuses more on core efficiency rather than flashy features. A lot of public tools rely on high-level frameworks like C#,

which are great for rapid development and heavy UI, but they naturally introduce more bloat and dependency.

For this project, I really wanted to challenge myself to build something strictly minimal and stable. That's why I chose a native C++ approach with zero dependencies—to prioritize performance and stealth over unnecessary overhead.

Since a few members expressed curiosity about the current state of my project, SokaRemote Professional v3.0, I am writing this follow-up post to share the latest updates on its development.

Currently, the client-side stub architecture is completely finished. I am now focusing entirely on building and polishing the controller/server side using Delphi. My main objective at this stage is squashing minor bugs, optimizing network routines, and maximizing overall stability.

Someone previously requested some screenshots, so I would love to showcase the current interface and some core modules below.

This is the main panel where active connections are managed. When you right-click on a target, a comprehensive context menu appears with a wide range of features.

A major architectural highlight of SokaRemote is its Custom Dynamic Loading (Zero-Import Stub) engine. Thanks to this low-level design, even if I add hundreds of additional features to the framework, the client stub size remains rigidly locked at around 28 KB. It does not bloat because modules are dynamically structured and streamed. The server is designed to handle this seamlessly, making the agent extremely lightweight and stealthy.

2. Screen Capture & Remote Control Module

This is the Remote Desktop form. It stream-decodes incoming frame packets smoothly on a dedicated background thread to prevent UI freezing.

Real-time Status Monitoring: The status bar provides absolute transparency with dedicated panels showing current streaming status, Resolution (Res), compressed Size (KB), and real-time frame rates (FPS).

Interactive Control: It supports smooth keyboard capturing, mouse movement, click synchronization, and custom JPEG quality/monitor selection.

3. Advanced Process Manager (Tree Structure)

This module maps the target's running tasks. Instead of a messy flat list, it utilizes a clean 2-pass tree structure algorithm to clearly separate Parent and Child processes.

Granular Details: Displays precise Process Name, PID, Architecture (Arch), and full executable Path.

Control Actions: Features a built-in search filter and allows administrators to Terminate (Kill), Suspend (Pause), or Resume processes on the fly.

4. Interactive File Manager & Transfers

The File Manager is split into two major operational tabs: File Explorer and File Transfers.

File Explorer: Allows full multi-drive navigation with real-time Windows system icon mapping. It fully supports standard operational hotkeys (F5 for Refresh, Del for Delete, F2 for Rename, Ctrl+C/X/V, etc.).

File Transfers: Houses asynchronous multi-threaded uploading and downloading tasks. It displays transfer progress with custom-drawn green progress bars, real-time metrics, transfer volume, and calculated speeds. It also includes an option to "Browse Local Downloads" to open the local storage directory immediately via context menu.

5. Startup Manager

This form interacts directly with the target's registry run keys to manage persistence.

Categorized Views: It fetches and parses entries into designated scopes: `CurrentUser\Run`, `CurrentUser\RunOnce`, `LocalMachine\Run`, and `LocalMachine\RunOnce`.

Registry Management: Provides an intuitive GUI dialog to seamlessly Add new startup items, Edit existing values/paths, or completely Remove persistence entries.

Current Status

Overall, the core architecture is performing incredibly well and working exactly as intended. I am currently focusing on optimizing the server's memory management, tuning the socket handling, and resolving minor visual layout bugs.

I will keep updating this thread as development progresses toward a stable release. Feel free to leave any technical questions or feedback below!

Best regards.

Update: Private Beta Program (Limited Slots)

The development of SokaRemote Professional v3.0 has reached its final stage. The core architecture is rock-solid, module streaming is seamless, and the server-side memory management has been fully optimized for high-traffic environments.

To ensure peak performance and compatibility before the public release, I am opening a limited number of spots for the Private Beta Program.

Why the \$600 USD participation fee?

Building a tool with Zero CRT and Zero STL dependencies in Native C++ is a massive undertaking that trades developer convenience for absolute efficiency. While high-level frameworks rely on bloated dependencies, SokaRemote is forged from the metal up.

Think of it this way: most tools on the market are like heavy, feature-laden SUVs—they work, but they are loud, slow to turn, and leave a huge footprint. SokaRemote is like a carbon-fiber supercar. With a stub size locked under 30KB, it offers superior stealth, unmatched connection speed, and a level of stability that simply cannot be replicated by high-level wrappers. You are paying for the architectural precision, the lightweight footprint, and the uncompromising performance that defines this project.

What you will get:

Early Access: Immediate, exclusive access to the current build of SokaRemote Pro v3.0.

Priority Support: Direct 1-on-1 communication with me for bug reports and custom performance tuning.

2-Month Update Cycle: Full access to all hotfixes and feature expansions for the next 2 months. After this period, your build remains fully functional, but it will no longer receive automated updates.

Beta Participation Fee: \$600 USD

If you are a serious professional who values low-level stability, stealth, and elite-tier engineering, please feel free to send me a private message (PM) for more details on how to join.

Note: This is an exclusive opportunity for serious testers. My priority for this release is quality and stability over mass accessibility.

Best regards.

Looking for beta testers to help iron out bugs and test features. If you are interested in helping post here and I will DM you if accepted.

Good old Delphi. Why don't you try AlphaSkins?

OtterLover - September 12, 2026, 3:46 am

Is there an update log?

Xinfo - September 12, 2026, 3:50 am

(Sep 12, 2026 03:46 AM)OtterLover Wrote: [\[CITE\]](#) Is there an update log?

I'll update the main post with the final version once it's completed.?

OtterLover - September 12, 2026, 3:54 am

(Sep 12, 2026 03:50 AM)Xinfo Wrote: [\[CITE\]](#) I'll update the main post with the final version once it's completed.?

Yes Sir~

Transporter007 - September 12, 2026, 4:18 am

will you add HVNC ?