What are the thread safety requirements of HSTRING and BSTR?

devblogs.microsoft.com/oldnewthing/20250312-00

Ian Boyd March 12, 2025

Raymond Chen

Among the proliferation of string types are the `HSTRING` (represented in C++/WinRT as `winrt::hstring` and in C++/CX as `String^`) and the `BSTR`. What are the threading rules for these string types?

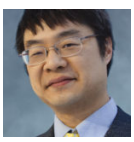
These string types are not COM objects, so the restrictions on COM objects do not apply. These are just blocks of memory that have freestanding functions for manipulating them (such as `SysAllocString` and `SysFreeString` for `BSTR`; `WindowsCreateString` and `WindowsDeleteString` for `HSTRING`). You are welcome to call any method from any thread.

The rules for `BSTR` are that read operations (like `SysGetStringLen`) can operate concurrently. But no read operations can operate concurrently with a write operation, so you cannot do a `SysGetStringLen` at the same time as a `SysReAllocString`, for example, and you can't modify the string contents on one thread while reading them from another.

Windows Runtime `HSTRING`s are immutable, so there are no write operations. This makes the rules simpler: Once you create an `HSTRING` (until you destroy it), all operations are thread-safe.

Author

Raymond Chen



Raymond has been involved in the evolution of Windows for more than 30 years. In 2003, he began a Web site known as The Old New Thing which has grown in popularity far beyond his wildest imagination, a development which still gives him the heebie-jeebies. The Web site spawned a book, coincidentally also titled The Old New Thing (Addison Wesley 2007). He occasionally appears on the Windows Dev Docs Twitter account to tell stories which convey no useful information.

2 comments

Discussion is closed. Login to edit/delete existing comments.

Newest

IB

Many years [ago][1], you mentioned that (as an implementation detail) the various shell/COM memory allocators were unified. In other words:

- CoTaskMemFree
- SHFree
- SHGetMalloc.Free
- CoGetMalloc.Free

are all the same thing behind the scenes.

Do we know if that extends to **BSTR** and SysFreeString?

I ask because Delphi has built-in language support for BSTR strings. In Delphi it a special type called **WideString**. The compiler calls SysAllocString and SysFreeString behind the scenes.

And there are quite a few times you want to call a shell function that returns a string, but it is pointer to string, and it must be freed using **CoTaskMemFree**....

Read more

KN

Kalle Niemitalo March 13, 2025
SysFreeString is not just a trivial wrapper around CoTaskMemFree.

* There is a BSTR cache; see the SetOaNoCache function.
* BSTR can have a reference count that prevents SysFreeString from freeing it immediately; see the SysAddRefString function.
* BSTR has a length stored at a negative offset, so the memory block that will have to be freed does not start at the address to which the BSTR points.

Stay informed

Get notified when new posts are published.