Google Talk for Developers

Overview

Open Communications

libjingle

- Developer Guide
  - Changelist
  - Important Concepts
  - How libjingle Applications Work
  - Creating a libjingle Application
    - Signing In to a Server
    - Sending and Querying Presence
    - Set Up the Session Management Pathway
    - Making and Receiving Connections
    - Scenarios
- Sample Applications
  - File Share Application
  - Voice Chat Application
- Reference
  - BasicPortAllocator Class
  - Call Class
  - Codec Struct
  - ChannelManager

MemoryStream Class

An extension of the base **StreamInterface** class designed for transferring information to another running application that can share a handle to this object (a sort of memory buffer object). It does not support **StreamInterface::SignalEvent** asynchronous notification. It creates an internal buffer to hold data to read and write to the stream. This subclass disables its copy constructor. You could use this interface to share data with an object that requires in-memory processing for speed or security (for example, an encryption or decryption class). The methods shown are not thread-safe, and can be called on any thread.

Syntax

```
class MemoryStream : public StreamInterface
```

Methods

Name	Description
void **Close**()	Unimplemented method, because no resources can be closed.
char* **GetBuffer**()	Retrieves the internal buffer that holds data in the stream. This buffer may change when **Write** is called.
const char* **GetBuffer**() const	A constant version of **GetBuffer**.
bool **GetPosition**(size_t **position*) const	Retrieves the current read/write position in the buffer. Returns True if successful, False otherwise.
bool **GetSize**(size_t **size*) const	Retrieves the size of the data in the buffer, in bytes. Returns True if successful, False otherwise.
StreamState **GetState**()	Returns SS_OPEN (a local memory stream is always open).

MemoryStream()
MemoryStream(const char* *data*)
MemoryStream(const char* *data*, size_t *length*)
	Constructor. Overloads with parameters enable the caller to specify initial data, as well as size of data (if known) in bytes.

~MemoryStream	Destructor. Deletes any memory resources allocated for its buffer.

Class
Connection Class
FileStream Class
HttpPortAllocator Class
MediaChannel Class
MediaEngine Class
MemoryStream Class
P2PTransportCha…
Port Class
PresenceOutTask Class
PresencePushTask Class
Session Class
SessionClient Class
SessionManager Class
SessionSendTask Class
StreamEvent Enumeration
StreamInterface Class
StreamResult Enumeration
StreamState Enumeration
Task Class
VoiceChannel Class
XmppClient Class
XmppEngine Class
XmppPump
XmppTask Class
License
Google Talk XMPP Extensions
User Settings

StreamResult **Read**(void **buffer*, size_t *bytes*, size_t **bytes_read*, int **error*)	Copies data from the internal buffer into the caller-supplied buffer. If the buffer supplied is too small, it returns an error. • *buffer* [in] A buffer containing the data to read. • *bytes* [in] The amount of data to read from the buffer, in bytes. • *bytes_read* [out] The amount of data read, in bytes. • *error* [out] EOF if the supplied buffer was too small to hold the requested data. • Return values: SR_EOS if the supplied buffer was too small to hold the requested data; SR_SUCCESS otherwise.
bool **Rewind**()	Sets the read/write position to zero and returns True for success, False otherwise.
bool **SetPosition**(size_t *position*)	Sets the read/write position to byte number *position* and returns True for success, False otherwise.
StreamResult **Write**(const void **buffer*, size_t *bytes*, size_t **bytes_written*, int **error*)	Copies the supplied data into the object. If the internal buffer is not large enough, it allocates data in blocks of 256 bytes. • *buffer* [in] A buffer containing the data to copy into the object. • *bytes* [in] The amount of data to write, in bytes. • *bytes_read* [out] The amount of data written, in bytes. • *error* [out] ENOMEN if memory could not be allocated. • Return values: SR_ERROR if memory could not be allocated; SR_SUCCESS otherwise.
bool **ReserveSize**(size_t *size*)	Allocates *size* bytes of memory for the stream.

> **Attributes:** public

> **Declaration file:** talk/base/stream.h

- Off the Record Chats
- Jingle Server Discovery
- Gmail Notifications
- Shared Status Messages
- Extended Contact Attributes
- JID Domain Discovery
- Voicemail
- OAuth 2.0 Authorization

Additional Resources

- Other IM Clients
- Accessories
- Google Talk Blog

Last updated March 23, 2012.

Search ●

Follow 1.7M

Explore

Products
Showcase
Events
Communities

Connect

Blog
Google+ Community
YouTube Channel
Report an Issue
Jobs

Programs

Groups
Experts
Startups
Women Techmakers

Top Products

Ads
Analytics
Android
Apps
Cast
Chrome
Cloud
Glass
Google+
Maps
Wallet
YouTube

Terms of Use | Privacy Policy

English ▼