??
?
??
$_{c}lass_{s}implified.pngCoremodulesimplifiedclassdiagram$
<
<
<
?
$_{t}>$
$[x][y] =$
$value|.However, thisisnowconsideredabaddesignchoice.Templating|Frame|forcedtheusertoknowthetypeofthedatabef$
?
??
$_{s}tructure]std ::$
$vector <$
$Field >$
$fields =$
$Field("x", Dtype(" < i4"), Field :: Scalar), Field("y", Dtype(" < i4"), Field :: Scalar), Field("pixels", Dtype(" < i4"),$
??
$_{S}IZE >$
$[hasbeenintroduced.$
$_{c}lusters]structClusterint32_{t}x; int32_{t}y; int32_{t}pixels[9];; //3x3clusterstd ::$
$vector <$
$Cluster >$
$clusters(10); //listof10clustersstoredcontiguouslyinmemoryfread(clusters.data(), sizeof(Cluster), clusters.size(), fil$
$_{i}o_{c}lass.pngFileIOmoduleclassdiagram$
??
??
??
$_{f}rame]//readingaFramefromafileFilefile1("data.npy"); Frameframe =$
$file.read();$
$_{t}Filefile2(path, "w", config); //openfileinwritemodefile2.write(frame); //writetheframetothefile//fileisclosedwhe$
??
$_{c}luster]//readingaClusterfromafileClusterFile <$
$ClusterHeader, ClusterData >$
$file("data.clust2"); autoresult =$
$file.read(); //read1recordfromfileClusterHeaderheader =$
$result.header; std ::$
$vector <$
$ClusterData >$
$clusters =$
$result.data;$
$_{f}ields.push_{b}ack("frame_{n}umber", Dtype :: INT32, Field :: NOT_{A}RRAY); header.header_{f}ields.push_{b}ack("n_{c}lusters", I$
$_{f}ields.push_{b}ack("x", Dtype :: INT16, Field :: NOT_{A}RRAY); header.data_{f}ields.push_{b}ack("y", Dtype :: INT16, Field ::$
$ClusterHeader, ClusterData >$
$file("file.clust2", "w", header);$
??

Network IO Module Implementation The network_io module is responsible for sending and receiving Frame objects

[width=]Chapitre3/figures/network$_{i}o_{c}lass.pngNetworkIOmoduleclassdiagram$

Figure ?? shows the class diagram of the network_io module.

ZeroMQ The network_io module uses ZeroMQ as the network library. ZeroMQ is a high-performance asynchronous

ZeroMQ's model provides socket communication between threads, processes and network nodes. Its sockets can wo

ZeroMQ's philosophy is to provide a simple API that can be used to build complex systems. It is very flexible and

ZmqSocket classes As seen from the class diagram, the network_io module implements ZmqSocketReceiver and Zmq

The ZmqSocket class follows the RAII idiom and wraps around the ZeroMQ socket. The ZmqSocketReceiver and Z

The network_io module methods return a ZmqFrame object. It is a wrapper around Frame that adds network meta

[language=C++, caption=Example of using the network_io module, label=lst:network$_{i}o_{e}xample]ZmqSocketRecei$
Example ?? shows how to use the network_io module to send and receive frames. The RAII idiom as always comes

ZmqMultiReceiver class In real world scenarios