DCMTK - Feature #180

Interprozesskommunikation in DCMTK (IPC)

2002-11-25 00:00 - Marco Eichelberg

Status:	Closed	**Start date:**	
Priority:	Low	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0:00 hour
Target version:			
Module:	ofipc ?	**Compiler:**	
Operating System:			

Description

hierzu hatten wir vor längerer Zeit eine Diskussion, und ich habe auch
mal angefangen, eine API zu implementieren, was aber nicht fertig geworden ist.

Die Frage ist, wie man einen allgemeinen, plattformunabhängigen Mechanismus
schaffen kann, um zwischen Prozessen Daten auszutauschen; basierend auf * Shared Memory * Semaphore, Mutex usw. * Signale (nur Unix) * Unix Domain Sockets * Named Pipes (nur Win32?) * ...

Es gibt eine abstrakte Nachrichtenklasse: class OFIpcMessage: * virtual OFCondition encode(stream) = 0; * virtual OFCondition decode(stream, Uint32 len) = 0; * virtual Uint32 typeInfo() const = 0;\\ [Nachrichtentypen: static const Uint32 OMT_ShutDown = 1; ...] * virtual Uint32 length() const = 0; * virtual OFIpcMessage *clone() const = 0;

Davon werden dann die konkreten Nachrichtenklassen abgeleitet.

Das Nachrichtenformat, das über den Socket geschickt wird, sieht dann so
aus (big endian byte order): * 0: Uint32 type; * 4: Uint32 msgID; * 8: Uint32 len; * 12: Uint8 payload[len];

Grundsätzliche Fragen zur Art der Kommunikation:

1) Richtung: * 1 Port pro Richtung * 1 Port für beide Richtungen
2) Ordnung: * point-to-point (1:1) * multi-point-to-point (m:1) * point-to-multi-point (1:n) * multi-point-to-multi-point (m:n)
3) Lebensdauer: * permanent * pro Nachrichtenpaar (RQ/RSP)
4) Threads: * multi * single + signal

History

#1 - 2025-08-21 14:40 - Marco Eichelberg

- Description updated

- Status changed from New to Closed

- % Done changed from 0 to 100

IPC ist inzwischen implementiert als class OFIPCMessageQueueServer.

#2 - 2025-08-21 19:51 - Michael Onken

Very cool :-)